Stylistic Structures

A Computational Approach to Text Classification

by Richard S. Forsyth

[Thesis submitted in accordance with the regulations governing the degree of Doctor of Philosophy in the Faculty of Science at the University of Nottingham, October 1995.]

To

Mei Lynn

Acknowledgements

Firstly I would like to thank the `Three Daves' for help both extensive and intensive: my supervisors David Clarke and David Elliman, and David Holmes (my guide into the world of authorship studies). They have each provided assistance and intellectual stimulation over and above the call of duty.

Many other people have helped in a variety of ways in the work leading up to this thesis. They include:

Nigel Allinson
Helen Forsyth
James Forsyth
Alex Gammerman
Paul Goodwin
Jim Gowers
Ken Jukes
Colin Martindale
Dean McKenzie
Chris Naylor
Richard Spencer-Smith
Alison Truelove
Fiona Tweedie
Richard Wright

Cordial thanks are also due to the inter-library-loan staff at the Bolland Library of the University of the West of England (U.W.E.). In addition, this project has benefitted from access to two public-domain text repositories, Project Gutenberg and the Oxford Text Archive, as well as from computing support provided by the Faculty of Computer Science and Mathematics at U.W.E.

Last but not least, I would also like to thank my wife, Mei Lynn, for support, suggestions and encouragement during the long period that this thesis took up, my daughter, Frances, for advice that altered the course of this research, and my son, Edward, for putting up with a good deal of thesis-induced absence and absent-mindedness.

Naturally, none of those named above should be held responsible for any blunders or blemishes in what follows.

Table of CONTENTS

Acknowledgements iii

Contents List iv

Abstract vi

1. Introduction 1
   1.1 Aims of this Research 2
   1.2 An Approach to Text Classification 4
   1.3 What this Thesis Isn't 5
   1.4 Preview of Subsequent Chapters 5

2. Background 8
   2.1 The Case of the *Federalist* Papers 9
   2.2 Other Stylometric Studies 20
   2.3 Other Relevant Research 42
   2.4 Review 49

3. Text Classification by Cross-Codebook Compression 56
   3.1 Why Data Compression? 56
   3.2 The Algorithm Used 57
   3.3 Some Results 60
   3.4 Critique 71

4. Text Classification with Diagnostic Digrams 77
   4.1 Towards an Empirical Framework 78
   4.2 Selection of Test Problems 79
   4.3 Preprocessing and Other Preliminaries 85
   4.4 Establishing a Baseline 87
   4.5 Trials on Word-Coded Data 95
   4.6 Results Using Word Transitions 101
   4.7 Discussion 103

5. Instance-Based Text Classification 105
   5.1 Text Classification by Similarity 105
   5.2 IOGA : An Instance-Oriented Genetic Algorithm 116
   5.3 From Prototypes to Profiles 125
   5.4 Concluding Comments 131

6. Text Classification by Rule Induction 133

6.1 A Robust Bayesian Text Classifier 133

6.2 Text Classification with Discrimination Trees 146

6.3 The Characterization of Character Strings 160

6.4 Review and Summary 162

7. Extending Textual Fragments to their `Natural' Lengths 165

7.1 How Long is a Piece of Substring? 165

7.2 Performance of Extended Text Fragments 173

8. An Evolutionary Approach to Text Classification 176

8.1 TOGA : A Text-Oriented Genetic Algorithm 176

8.2 Preliminary Test on Numeric Data 191

8.3 Testing Various Fitness Functions 198

8.4 An Experiment on Rule Complexity and Inference Mode 210

8.5 Does Stricter Selection of Features Help? 215

8.6 Evaluative Review 221

9. The *Federalist* Revisited (Again): a Case Study 227

9.1 A Break from Benchmarking 227

9.2 A Stepwise Removal of Restrictions 236

9.3 Though this be Madison, yet there's Method in it 239

9.4 Summing Up 243

10. Concluding Discussion 245

10.1 Achievements 245

10.2 Findings 254

10.3 Failings and Future Prospects 259

10.4 Closing Remarks 267

Reference List 271

Appendix A -- Specimen Spitbol Program 1: A Monte-Carlo Feature Finder A-1

Appendix B -- Specimen Spitbol Program 2: Basic Bayesian Text Classifier B-1

Appendix C -- Specimen C Code: GA Routines and Heap-Tree Functions C-1

Appendix D -- Details of Data Sources D-1

Abstract

The problem of authorship attribution has received attention both in the academic world (e.g. did Shakespeare or Marlowe write *Edward III*?) and outside (e.g. is this confession really the words of the accused or was it made up by someone else?). Previous studies by statisticians and literary scholars have sought "verbal habits" that characterize particular authors consistently. By and large, this has meant looking for distinctive rates of usage of specific marker words -- as in the classic study by Mosteller and Wallace of the *Federalist Papers*.

The present study is based on the premiss that authorship attribution is just one type of text classification and that advances in this area can be made by applying and adapting techniques from the field of machine learning.

Five different trainable text-classification systems are described, which differ from current stylometric practice in a number of ways, in particular by using a wider variety of marker patterns than customary and by seeking such markers automatically, without being told what to look for. A comparison of the strengths and weaknesses of these systems, when tested on a representative range of text-classification problems, confirms the importance of paying more attention than usual to alternative methods of **representing** distinctive differences between types of text.

The thesis concludes with suggestions on how to make further progress towards the goal of a fully automatic, trainable text-classification system.

Chapter 1

INTRODUCTION

> "As we near the end of the twentieth century, the printed book also appears to be drawing near the end of its five-century career." -- Philip Davies Roberts (1986).

The world is awash with text.

When Saint Alcuin (c.732-804 A.D.) was librarian at York Minster he presided over the revival of learning known as the "Northumbrian Renaissance" (Hutchinson & Palliser, 1980). Before being destroyed by the Vikings in 866, Alcuin's library held a collection of about 300 manuscript volumes, estimated as almost half the books in Europe at that time (Allinson, 1994). In modern terms, allowing 60,000 words per book and using 6 bytes as a rough estimate of storing each word in a computer memory, that comes to 108 megabytes. Were he alive today, Alcuin could equip himself with a portable laptop computer that could hold three or four copies of his entire library -- without using any form of data compression.

Alcuin flourished during the so-called Dark Ages, notorious as a low point in European scholarship; but even the fabled library of Alexandria, the greatest treasury of literature assembled at the height of classical civilization, is said to have contained no more than half a million papyrus scrolls (Sagan, 1981). As a scroll typically held about 10,000 words, that comprises around 5 billion words or 30 gigabytes of storage -- too much for you or me or Alcuin to carry around in a briefcase, but still not huge by today's standards. For comparison, Leech (1987) states that the text databank of the Mead Data Corporation contained 5 billion words available for on-line retrieval in the mid-1980s. It was already the size of the great library of Alexandria, and has undoubtedly grown since then.

There is, therefore, plenty of text in our industrialized society, much of it stored on computers. Indeed there is so much text being stored on, and transmitted between, computers (as electronic mail, faxes, word-processed documents and so on) that we suffer from text overload. Individuals, and indeed organizations, cannot really cope with such vast amounts of textual information. In other words, our ability to analyze text lags well behind our ability to save, retrieve and send it. This mismatch between ability to store and ability to analyze texts has motivated responses of several kinds -- including, for example, the development of programs that filter out `junk email'.

The work reported in this thesis can also be seen as a response to this predicament.

1.1 Aims of this Research

The most important objective of this thesis (and the author's excuse for adding to the mountain of text described above) is to make a modest contribution to the field of text analysis.

By `text analysis' is meant any method of summarizing or extracting information from data that is encoded solely or chiefly in the form of character strings. There are no departments of Text Analysis, so relevant work is currently carried out under several headings, including computer science, linguistics, psychology and statistics. In the English-speaking world the term text analysis does not even designate a well-defined area; although in France it does seem to be emerging as a coherent field (see, for example, Lebart et al., 1991; Lebart & Salem, 1994). It is hoped that, after reading this thesis, the reader will share the author's view that text analysis deserves to be considered a worthwhile field of study in its own right.

To be more specific, the present work concentrates on methods of text classification, where the main aim is to find ways of using patterns within texts to sort them as accurately as possible into classes or groups. In fact, much of what follows will be concerned with the subproblem of authorship attribution, i.e. how to decide who wrote a piece of text of unknown or disputed provenance. But this focus on authorship attribution is mainly to keep the task manageable; and occasional forays will be made into other types of text classification (such as whether a text was written during an author's earlier or later years) to ensure that the methods developed can, at least potentially, be used on a wide range of text categorization problems.

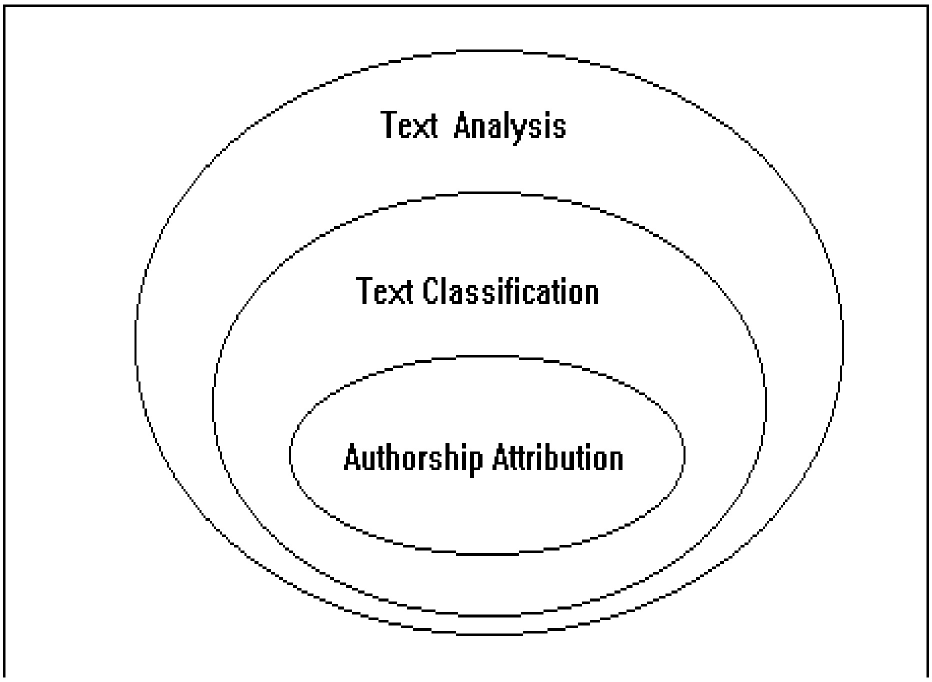


Figure 1.1 -- Subfields of Text Analysis.

Figure 1.1 is meant to clarify the position of the present study by locating it within an appropriate conceptual space. This thesis is concerned mostly with authorship attribution, but that is only one kind of text categorization. Other example problems in text categorization are:

- [clinical example] does this transcript come from a patient with damage to Broca's area of the left hemisphere or some other region of the brain?
- [financial example] was this newspaper article written about a company that filed for Chapter 11 bankruptcy within nine months of the story being published or not?

(It will be seen that if an effective general-purpose text classification system could be developed it would have many useful applications.)

There are also problems of text analysis other than classification. These include clustering. For instance: does a set of texts fall naturally into distinct clusters and, if so, how many? If the texts concerned are all written by the same author, this amounts to asking how many different genres can be identified in that author's work. However, there is insufficient room here to give such tasks, interesting as they are, more than a passing mention.

The present research is concerned primarily with the innermost oval of Figure 1.1, although with an awareness that methods developed for authorship attribution should ideally be easy to generalize to other text classification tasks (the middle oval).

1.2 An Approach to Text Classification

Another way of elucidating the main purpose of this thesis is to state that the project which it describes is intended to advance towards a long-term goal. That goal is the development of a computer-based text classification system that differs from current practice in three main respects:

(1) by using marker patterns that are not necessarily words;
(2) by finding them automatically, without being told what to look for, and without relying on knowledge sources external to the texts being analyzed;
(3) by paying more attention to the sequential structure of text than is customary.

The principal objective of this project is to make progress towards that goal, even though it is not fully attained by the final chapter. (The reasons for emphasizing the three features listed above will be outlined in Chapter 2.)

The rest of this document describes a number of related attempts to achieve that objective, employing a variety of techniques (some of them novel, at least as applied in this field). It also describes various items of software developed as part of the overall project and some findings resulting from testing these software systems. It is the author's contention, however, that the main outcome of this research is not so much the specific results reported nor the techniques employed in uncovering them nor the software tools written to support those techniques, nor even a general framework into which such techniques, and software supporting them, can be embedded (though suggestions will be made on these scores). Rather, the main outcome of this work is that it makes a certain amount of progress towards answering a question:

➢ what kinds of representation are most useful for characterizing differences between texts?

Thus, as the title *Stylistic Structures* suggests, a principal focus of this work is on proposing and evaluating alternative ways of representing distinctive differences among texts. As will become clear in Chapter 2, previous researchers in authorship studies and allied fields have tended not to ask this question and so to answer it only implicitly. Bringing it into the foreground, it is hoped, will open up several fruitful avenues that might otherwise remain unexplored.

1.3 What this Thesis Isn't

Having briefly outlined the aims of the present work, it is as well to add a few words here concerning what it does not attempt to achieve.

This research does not really belong in the main stream of computational linguistics or natural language processing as those terms are commonly understood; although it clearly touches the fringes of those subjects. According to Grishman (1986), "computational linguistics is the study of computer systems for understanding and generating natural language". Computational linguists tend to spend most of their time tackling two difficult problems: (1) parsing natural-language text, (2) translating the parsed text into some form of internal representation which is more amenable than free text to automatic inferencing procedures. Simply in order to classify texts, however, parsing is not essential -- still less understanding. So none of the systems described herein is part of an attempt to understand natural language input. No semantic analysis is performed, and very little syntactic analysis in the usual sense of that term.

This means that the present work can only claim to be a contribution to the field of Artificial Intelligence (AI) if that term is interpreted more broadly than perhaps is normal; even though it borrows some ideas from AI and related areas, e.g. Machine Learning and Pattern Recognition, and (it will be argued) in the process of adapting them to textual problems adds to our knowledge of their strengths and weaknesses.

Nor does it offer much of help to designers of word-processing or desktop publishing systems -- the two chief kinds of practical text-processing in the commercial world.

(In spite of this disclaimer, it will be suggested in the final chapter that the sort of work described in this thesis does have practical relevance to some adjoining disciplines, such as Information Retrieval.)

1.4 Preview of Subsequent Chapters

To help readers make use of this document, this section gives a short outline of what to expect in the following chapters.

Chapter 2 contains a brief survey of previous research in authorship studies and related fields. In this survey the work of Mosteller and Wallace (1984) is given special prominence both because it is a very significant piece of research in its own right and because it serves as a good introduction the field of stylometry, which is the discipline in which most authorship attribution problems have been studied to date.

Chapter 3 introduces the concept of cross-codebook compression during an investigation into whether a simple method of data compression can be adapted to become a workable text-classification technique. The attempt proves only partly successful, and thus Chapter 3 might be regarded as something of a digression, not centrally relevant to the main thrust of this research; however, it is included because it introduces a theme that recurs throughout this thesis -- the use of short substrings as diagnostic features -- and also because many of the developments in later chapters are most easily understood as ways of rectifying the failings of cross-codebook compression.

Chapter 4 describes a simple text-classification program, written in a dialect of Snobol4, called BBTC (the Basic Bayesian Text Classifier). BBTC uses digrams, trigrams or tetragrams to make probabilistic categorizations of unseen text samples. Though simple, this program establishes a baseline performance level against which the more sophisticated trainable text classifiers of later chapters can be compared. In addition, since such comparisons require some sort of empirical grounding, this chapter introduces a benchmark suite of 13 text-classification problems (spanning a wider range of domains than in most published stylometric studies) which will serve as a yardstick for evaluating the methods tested in the following four chapters.

Chapter 5 reports results obtained by applying both a standard and a novel method of instance-based classification to the 13 benchmark problems. As instance-based classifiers have been found to work, in general, rather well on non-textual problems (Aha et al., 1991; Fogarty, 1992; Michie et al., 1994) this provides a link between the present research (primarily concerned with textual data) and the wider field of Machine Learning (primarily concerned with data encoded as numeric feature vectors), thus enabling external as well as internal comparisons to be made.

Chapter 6 assesses two quite different trainable text classifiers -- one using a genetic algorithm to optimize a Bayesian classifier, the other based on the widely used machine-learning technique of inducing discrimination trees. Results on the benchmark problems suggest that the former system is more promising, even when an efficient tree-pruning algorithm is used.

Chapter 7 focuses on extending and improving the software for what is termed here Monte-Carlo Feature-Finding, thus somewhat refining a technique of discovering distinctive substrings that emerged almost as an accidental by-product of the work in Chapters 5 & 6 (but which, with hindsight, may well prove to be the most valuable contribution of this whole thesis).

Chapter 8 recounts a further refinement to the Monte-Carlo feature-finding software; but it is mainly a description of a suite of programs that constitute an evolutionary text-oriented inductive classifier. This is an attempt to consolidate, into a single (`state-of-the-art') system, lessons learned from work described in Chapters 3-7, especially Chapters 5-7. This is also the chapter that pays most serious attention to the problem referenced in the title *Stylistic Structures*, namely the problem of devising a good <u>description language</u> for representing in a compact and comparatively comprehensible form induced knowledge about how to distinguish between different types of text.

Chapter 9 winds up the substantive work of this project with a small-scale case study of a well-known hard authorship-attribution problem, and lays down some guidelines for effective practical use of the software systems developed in Chapters 6-8.

Chapter 10 reviews the findings of earlier chapters and assesses the strengths and weaknesses of the systems tested therein, thereby endeavouring to evaluate the contribution made by the work that this thesis describes. In the process it also attempts to reach some generally valid conclusions about automated text categorization and to point to some areas where further research may be expected to be beneficial.

Chapter 2

BACKGROUND

> "Anything a person writes contains the code of his intellectual DNA, or whatever you want to call it." -- Charles Webb (1994).

This chapter reviews some of the more important work relevant to text classification carried out over the past 30 years. Most of the studies reported come from the field known as stylometry, which can be defined, briefly, as the search for verbal features characterizing certain styles that are both measurable and stable over time. In the words of Matthews & Merriam (1993):

> "Stylometry attempts to capture quantitatively the essence of an individual's use of language. To do this, researchers have proposed a wide variety of linguistic parameters (e.g. rare word frequencies or ratios of common word usage) which are claimed to enable differences between individual writing styles to be quantitatively determined."

The stylometric problem that has received most attention in the literature is the problem of authorship in cases of doubtful or disputed attribution. This is because, although the word `style' has many senses, it is most often used to refer to an author's individual quirks of phrasing or vocabulary. As Middleton Murry (1960) puts it:

> "Style naturally comes to be applied to a writer's idiosyncrasy, because style is the direct expression of an individual mode of experience."

Attempts to settle questions of disputed authorship have been, and remain, important not only in the academic world (e.g. did Shakespeare or Fletcher write the play *Henry VIII*?) but also outside, especially in forensic contexts (e.g. is this confession an accurate record of a statement made by the accused or was it made up by someone else?). Thus the bulk of this chapter will be concerned with authorship attribution, although we will look at a few examples of other kinds of text discrimination.

It should be noted that all stylometric studies of authorship rest on a basic assumption, namely that any author will be found, given sufficient analytic scrutiny, to have distinctive and consistent verbal habits. This basic assumption has been articulated by, among others, Morton (1965):

> "prose writers have stylistic habits which persist over wide ranges of subject matter and throughout long periods of time. For works in the same literary form, the differences between works are only expected differences of random sampling."

Morton believed he had proved this assertion, at least with regard to a variety of ancient Greek authors; but, strictly speaking, it is safer to regard it still as a conjecture, albeit a conjecture that has received a good deal of empirical support in the years since Morton's investigations of classical Greek prose. Holmes (1985) puts the same point less tendentiously as follows:

> "Once it has been established what texts are to be compared, the stylistic analyst has to determine whether the variance within a text is larger or smaller than that between texts, i.e. the analyst has to show that what is being measured varies more between works of two authors than between different works of the same author."

Here it is clear that the existence of distinctive authorship markers cannot be taken for granted, but must be established anew for each author studied. In the rest of this chapter we look at some ways in which various investigators have attempted to find such linguistic markers.

The origins of quantitative stylometry can be traced back over more than a century (see Lord, 1958), but the subject was not put onto a firm scientific footing until Yule (1944) in *The Statistical Study of Literary Vocabulary* introduced many of the techniques still in use today. However, in the interests of brevity, we will pass over the early days of literary statistics -- referring the interested reader to excellent summaries by Williams (1970) and by Oakman (1980) -- and begin this survey with a relatively detailed look at the work of Mosteller & Wallace (1964, 1984) on *The Federalist Papers*. This thorough and meticulous study, deservedly regarded as a classic of its kind, serves as an excellent introduction to the field, even 30 years after its initial publication.

2.1 The Case of the Federalist Papers

During 1787 and the spring of 1788, 77 articles were printed in four of New York

City's five newspapers, with the aim of persuading New Yorkers to support ratification of the proposed new constitution of the United States of America. These articles appeared over the pseudonym Publius and, as it happens, were unsuccessful: 56% of the citizens of New York state voted against ratifying the constitution. Undeterred by this setback, Publius re-issued these propaganda pieces in book form in May 1788, together with an additional eight essays that had not previously been published, so that delegates at the Constitutional Convention, then sitting, might be swayed by their case in favour of federalism. The New York delegation did eventually abandon opposition to the constitution, but mainly, it is thought, because nine of the thirteen states ratified, leaving New York potentially isolated (Wills, 1982). The book, however, has remained in print for over 200 years.

Speculation concerning the identity of Publius was widespread at the time, and gradually it became accepted that General Alexander Hamilton had been heavily involved in the composition of the Federalist papers but that he had not written them all alone. Hamilton died in a duel with Aaron Burr in 1804, and in 1807 a Philadelphia periodical received a list, said to have been made by Hamilton just before his fatal duel, assigning specific papers to specific authors -- himself, John Jay and James Madison (the fourth president of the United States). Not until he retired from the presidency did Madison concern himself with contesting Hamilton's account of who wrote particular *Federalist* papers, but in 1818 he claimed full authorship of numbers 49-58 as well as 62 and 63 -- papers that Hamilton's list had ascribed to himself. Thus 12 of the 85 papers were claimed by both Hamilton and Madison.

From that time till 1964, when Mosteller and Wallace published the first edition of their book, entitled *Inference and Disputed Authorship: the Federalist*, scholarly opinion was divided. Both Hamiltonian and Madisonian authorship of the 12 disputed papers were seriously argued on a variety of historical and stylistic grounds. For reference, the situation that faced Mosteller and Wallace at the outset of their study, with regard to authorship of particular numbers of *The Federalist* is tabulated below.

Table 2.1 -- Accepted Federalist Authorship in 1964.

Paper Number(s)	Author
1	Hamilton
2-5	Jay[1]
6-9	Hamilton
10	Madison
11-13	Hamilton
14	Madison
15-17	Hamilton
18-20	Joint: Hamilton & Madison
21-36	Hamilton
37-48	Madison
49-58	** Disputed
59-61	Hamilton
62-63	** Disputed
64	Jay
65-85 Hamilton	

In the (extended) 1984 edition of their monograph, Mosteller and Wallace report at least 6 studies bearing on the question of who wrote these 12 disputed papers:

(1) a preliminary study involving a linear discriminant function;
(2) "the main study";
(3) various sensitivity analyses of the main study;
(4) a "weight-rate analysis";
(5) a "robust hand-calculated Bayesian analysis";
(6) a "three-category analysis".

They also describe some other subsidiary investigations of *Federalist* and other themes; but we will concentrate on the six items above, particularly what they call "the main study".

2.1.1 The Pilot Study

Unpublished work by Frederick Williams and Mosteller had convinced them that

[1] John Jay was wounded in a street riot soon after the series was launched, which is why his contribution is much smaller than Hamilton's or Madison's.

sentence length was a poor discriminator. Indeed Hamilton and Madison are almost identical on this measure, as Table 2.2 shows.

Table 2.2 -- Sentence Length of Hamilton and Madison.

	Hamilton	Madison
Mean sentence-length in words =	34.55	34.59
Standard deviation =	19.2	20.3

An initial glimmer of success came in a pilot study using a linear discriminant function based on the following variables: proportion of nouns; proportion of adjectives; number of 1-letter and 2-letter words; frequency of the definite article. A discriminant function derived from undisputed papers by Hamilton and Madison assigned nine of the disputed papers to Madison and three to Hamilton.

This result was suggestive rather than convincing, so Mosteller and Wallace turned their attention to what they call "marker words", having been informed that `while' and `whilst' were used with very different frequencies by the two authors and having noticed that `upon' and `on' behaved in a similar fashion. This is shown in Table 2.3. [Source: Mosteller & Wallace (1972).]

Table 2.3 -- Hamiltonian and Madisonian Marker Words.

Rate per Thousand Words of:	On	Upon	While	Whilst
Hamilton	3.38	3.24	0.21	0.08
Madison	7.75	0.23	0.07	0.42

2.1.2 The Main Study

In their main study, Mosteller and Wallace used Bayes's Theorem to draw inferences about the probabilities of the competing hypotheses that Hamilton or that Madison wrote the disputed *Federalist* papers, based on evidence provided by rates of usage of 30 marker words, including the four shown above in Table 2.3. They concluded that Madison was very likely the author of all 12 disputed papers.

Such a bald summary, however, does less than justice to what was a considerable intellectual feat, requiring much ingenuity as well as hard work.

First they collected a large amount of text from undisputed works by both authors, 94,000 words written by Hamilton and 114,000 words by Madison. It should be noted that some of these writings were from outside *The Federalist*: less than 10,000 words of their Hamilton sample were from non-Federalist sources but in Madison's case the majority, almost 73,000 words, was from other works of his, composed over a 25-year period. This fact necessitated a subsidiary study of Madison's literary output to make sure that his rate of usage of the kind of words being counted (mostly high-frequency function words such as `any', `by', `from', `there' and so on) was indeed stable over time.

The next step involved choosing suitable marker words. This was done in several sweeps through subsets of the texts, which with hindsight can be seen to fall into two main phases. Firstly, well over 300 potential marker words were tested against a set of screening texts of about 40,000 words by each author. Only 165 words, that showed significant ability to discriminate between texts by Hamilton and Madison, were retained. Then these 165 were whittled down to the final 30 by testing on a second set of texts that had not been used in the initial screening. The idea behind this two-stage procedure was to guard against overfitting.

At this point Mosteller and Wallace had their discriminators and could apply Bayes's Rule to make statistical inferences. This rule for amending probabilities in the light of evidence (Bayes, 1763) lies at the heart of the main study. Mosteller and Wallace used logarithms of odds rather than probabilities and took advantage of the fact that they were working with only two possibilities, so they were able to work with a simplified version of Bayes's Rule that can be expressed as:

Final log-odds = initial log-odds + log-likelihood.

Log-likelihoods for each marker word, derived from differential rates of usage, were simply added (assuming independence between words) to reach a final assessment.

However it is perhaps not generally appreciated that Mosteller and Wallace carried out in this study **two logically separate stages** within the overall Bayesian reasoning process, as illustrated by Figure 2.1, nor that the first stage -- arriving at appropriate log-likelihoods given particular word counts in papers of known origin -- involved much greater mathematical complexity than the second -- applying those log-likelihoods to decide the authorship question.

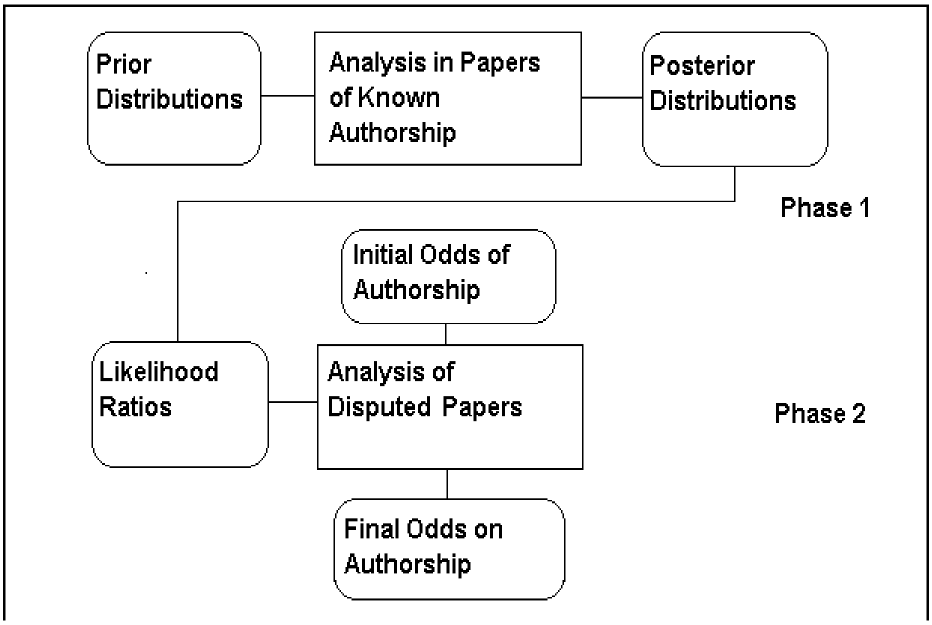


Figure 2.1 -- Two Phases of Bayesian Inference.

The need for an initial stage arose because even with quite large amounts of text and relatively common words, the observed mean rates of word usage were insufficiently accurate to be used as they stood. This can be illustrated by the word `also', which occurred at a rate of 0.28 per thousand words in Hamilton's sample, that is 26 times in 94,000 words. While 94,000 words is a large sample, 26 occurrences is not; and the standard error of the mean value 0.28 comes to 0.055 which is by no means negligible.

So Mosteller and Wallace studied the behaviour of 90 function words not chosen in the final pool of 30 markers to arrive at a realistic mathematical model of word usage behaviour. They concluded that, with few exceptions, word frequencies were badly modelled by assuming a Poisson distribution but that the negative binomial distribution gave good fits for the sort of words they were interested in. However each negative binomial model for each of 30 marker words required four parameters to be specified and in most cases there was not enough text simply to estimate these values from the data -- as explained above. To surmount this problem they treated these parameters as uncertain variables and applied Bayes's Rule to compute a `posterior' estimate of the probability distribution of their values using texts of known authorship. Of course, this in turn required a model of the prior distribution for such values: their investigation of the 90 non-marker words convinced Mosteller and Wallace that the Beta distribution was satisfactory for this purpose.

The practical effect of this preliminary Bayesian stage before the main authorship analysis was to moderate the difference between Hamiltonian and Madisonian markers, especially for relatively rare words. This they interpeted as giving further protection, additional to the double screening of potential markers, against any bias introduced by selecting only highly discriminatory words[2].

At the end of all this they were able to combine the evidence from 30 separate marker words and thus derive posterior log-odds in favour of Madison's authorship of the disputed papers. The weakest of the 12 resulting log-odds were those for paper 55, representing a multiplier of about 240 in favour of Madison, sufficient to overturn any but the most extreme prior odds on Hamilton.

As Hockey (1980) comments: "it is generally accepted that this particular authorship problem has been solved conclusively."

2.1.3 Sensitivity Analyses

Many supplementary analyses were also done in support of the main study. These can be bracketted under the heading `sensitivity analysis' as they were designed to test the sensitivity of the main conclusion (that Madison wrote all 12 disputed papers) to a range of different assumptions. Among the issues explored which might have affected the final conclusion were:

- choice of negative binominal versus Poisson model;
- sensitivity to a wide range of plausible priors for model parameters;
- importance of assumption of independence among words;
- size of regression effects.

To cut a long story short, they concluded that none of these sources of imprecision demanded more than minor adjustments to the calculated posterior odds on Madison's authorship.

2.1.4 A Weight-Rate Analysis

This analysis can be seen as an extension of the pilot study using a linear discriminant function (subsection 2.1.1) in the light of the main study's findings. It was performed largely to find out whether a classical, i.e. non-Bayesian, statistical analysis would

[2] They observed that using a `tight' prior distribution, such as the Beta distribution, with a somewhat heavy-tailed data distribution, contrasts with most applications of Bayes's Theorem, where a relatively `flat' prior distribution is typically modulated by a rather sharply-peaked data distribution, and concluded that doing it their way round had much to commend it.

support the conclusions of the main study.

Firstly a pool of 117 potentially discriminating words was tested against a screening set of 23 papers by Hamilton and 25 by Madison. Only the most discriminatory 20 words were retained. Of these, the best 10, in order of importance were:

`upon'	+	
`whilst'		-
`there'	+	
`on'		-
`while'	+	
`vigor'	+	
`by'		-
`consequently'		-
`would'	+	
`voice'		-

Here a plus sign indicates a word more often used by Hamilton, a minus sign a word more used by Madison.

For each of the 20 selected marker words a weight was calculated to form a linear discriminant function. This function could be applied to a paper of known or unknown authorship by multiplying the rate of occurrence of the 20 markers (scaled per thousand words) each by its weight and summing these products to give a numeric value, y', which was positive for Hamilton and negative for Madison. Actual numeric values of y' were arbitrary; unlike log-odds, they were merely indicators.

On the screening set, the mean value of y' was 0.87 for 23 Hamilton papers and -0.41 for 25 Madison papers. As these papers were used to select words and their weights, this consitutes a self-test. On a calibrating set of unseen papers, the mean values of y' were 0.92 for 25 Hamilton papers and -0.38 on 25 Madison papers. It turned out that there was no overlap on the screening set or, more importantly, on the calibrating set: the lowest-scoring Hamilton paper and the highest-scoring Madison paper in the calibrating set were about half a standard deviation apart; and although the variance of the y' values increased from screening to calibrating sets, the mean y' values of the two authors were still approximately 4.5 standard deviations apart.

Based on the means and standard deviations of y' values calculated from this calibration set, all twelve disputed papers except number 55 were outside the 99% confidence interval for Hamilton, while all but numbers 55 and 56 were inside the 90% confidence interval for Madison. Thus a non-Bayesian statistical analysis gave results that were very similar to those of the main, Bayesian, study -- although somewhat less conclusive.

Of the three joint papers, number 20 appeared more Hamiltonian than Madisonian on the basis of y' value.

A weakness of this weight-rate study is that no allowance was made for the effect of differing paper lengths.

2.1.5 The Robust Bayesian Analysis

In this study Mosteller and Wallace sacrificed information in order to avoid some of the difficulties connected with choosing a suitable family of prior distributions and estimating what have come to be known as `hyperparameters' for such distributions.

The problem of modelling word usages was simplified by dichotomizing rates of usage: common words were categorized into two classes, above or below the median rate (for both authors combined); less common words were also put into two classes, zero or non-zero rate of occurrence in a 2000-word block. Thus the information about each word could be encapsulated in a 2x2 table. (The payoff for this simplification was that the whole study could be executed using only slide rules.) An example, for the word `to' appears below.

Table 2.4 -- Fourfold Frequency Table for word `to'.

Rates compared to a cutoff rate of 37.805 per thousand words:	Low	High
Hamilton	7	16
Madison	16	7

These figures were obtained on a screening set of 46 papers, 23 by each author, chosen to be close to 2000 words in length. In fact the range of paper lengths was from 1728 to 2282 words. This table shows that in 7 of his 23 papers Hamilton used the word `to' less than 37.805 times per thousand words and in 16 papers more often than that -- vice versa for Madison.

Mosteller and Wallace tried 193 words on this screening set and whittled this pool down to 31 that showed ability to discriminate between Hamilton and Madison. For each of these 31 words, a 2x2 table like that above was obtained and used to compute appropriate likelihood factors. In fact these tables should be seen as nothing more than devices for computing odds multiplication factors.

To test the procedure, Mosteller and Wallace used the 31 tables to calculate posterior log-odds on Hamilton's authorship (assuming even prior odds between Hamilton and Madison) for each of the 46 screening papers. The mean posterior log-odds for the 23 Hamilton papers came to 13.95; for the 23 Madison papers it came to -14.25. On a validation set of 31 papers, not used to form the 2x2 tables, the mean log-odds were 10.2 for 13 Hamilton papers and -8.2 for 18 Madison papers. Thus, as expected, there was a shrinkage from screening to calibrating papers; but it is worth noting that even log-odds of 8.2 represent odds of 3641 to 1 in favour of the correct author.

When applied to the twelve disputed papers, this method gave mean posterior log-odds of -5.66. All were strongly Madisonian expect number 52 (posterior odds of 7 to 1 in favour of Madison) and number 55 (once again) where the resulting odds were 10 to 1 in favour of Hamilton.

2.1.6 A Three-Category Analysis

This study was very similar to the robust Bayesian study reported in the preceding subsection. It differed in only two significant respects:

(1) 3x2 tables (distinguishing Low, Medium and High rates of usage for both authors) were used, except in a few degenerate cases;
(2) log-odds were calculated in the same manner as described in 2.1.5 but were not given a probabilistic interpretation: they were merely used as arbitrary scores.

Slightly different sets of screening and validating papers were used and a slightly different pool of markers survived the screening process, compared with those described in subsections 2.1.4 and 2.1.5, but the procedure was in essence the same. In this case 63 of an initial pool of 117 words were retained after the screening.

Using posterior log-odds simply as scores, rather like the y' values of a regression formula used as a discriminant function (as in the weight-rate analysis), the separation between Hamilton and Madison mean values dropped from 7.3 standard deviations on the screening set of 23 Hamilton and 25 Madison papers to 3.6 standard deviations on a calibration set of 25 Hamilton and 25 Madison papers. Two `mistakes' were made on this validating set: one Hamilton paper fell on the Madisonian side of the midpoint and one Madison paper fell on the Hamiltonian side.

On the disputed papers, taking means and standard deviations from the validation set, all but number 55 (again) were on the Madisonian side of the midpoint. Seven of the 12 were more than 2 standard deviations from that midpoint in the Madison direction.

2.1.7 Appraisal

An attempt to evaluate the contribution of Mosteller and Wallace, together with other studies reviewed in this chapter, will form part of section 2.4. However its status as a definitive work of its type merits a provisional appraisal in advance.

First it is worth noting their own self-assessment (Mosteller & Wallace, 1984):

> "We tracked the problems of Bayesian analysis to their lair and solved the problem of the disputed *Federalist* papers."

As this ordering suggests, they were primarily interested in demonstrating a practical application of Bayesian statistics, and in solving some of the problems associated with doing so, and only secondarily in settling the question of who wrote the disputed *Federalist* papers.

A second point worth making at this stage is that they have been more admired than emulated. As will be seen in the rest of this chapter, subsequent researchers have been reluctant to follow them down the path they pioneered; and Bayesian reasoning has not played a major part in stylometric studies in the three decades since 1964. Indeed it is not much of an exaggeration to say that stylometrists have dropped the baton handed on by Mosteller and Wallace.

The reason for this cannot be identified with certainty, but my own view is that on the one hand humanist scholars found the extremely elaborate conceptual apparatus simply too hard to comprehend while, on the other hand, statisticians (who were at least able to understand most of the mathematics) still tend to regard Bayesian methods with suspicion. Even today statisticians mostly fight shy of anything that might involve a subjective interpretation of probability. The fact that Mosteller and Wallace obtained rather similar results with a non-Bayesian approach may have told against their advocacy of Bayesianism.

From a polemical point of view Mosteller and Wallace may thus be said to have failed in their main objective (rather like Hamilton and Madison before them!): they have impressed but not persuaded the majority of their intended audience. They might actually have had more followers if their work had been less comprehensive, for example if they had never published their book but only a paper on the robust Bayesian analysis, which is easier to understand and copy than the main study. (This is hardly best placed for maximum impact on readers: it lies buried as chapter 6 of 10.)

2.2 Other Stylometric Studies

In this section we consider, in chronological order, some of the more noteworthy studies of authorship and allied matters that have been carried out since the first edition of Mosteller and Wallace's book in 1964.

2.2.1 The Style of Dean Swift

Shortly after Mosteller and Wallace completed their study of *The Federalist Papers* a book was published (Milic, 1967) containing a quantitative analysis of the style of Jonathan Swift, best known as the author of *Gulliver's Travels*.

Having reviewed over 200 years' worth of highly impressionistic and subjective descriptions of Swift's style by various literary scholars, Milic concluded that very little was really known about that subject, and decided that the only remedy was a quantitative approach. To this end, he encoded 18 text samples of about 3500 words each on punched cards. Eight of these were from Swift's core canon; eight were by near-contemporaries of Swift (two each by Addison, Gibbon, Johnson and Macaulay); and two were the first and second half of a piece of just over 7000 words entitled *Letter of Advice to a Young Poet* which some but not all scholars believed to have been written by Swift.

Believing that writers reveal their individuality more in their syntax than their vocabularies, Milic did not analyze raw text but sequences of grammatical codes, punched onto cards.

> "To simplify the process and make only such use of the computer as might be efficient, I decided to analyze the texts into word-classes `manually'. The alternative -- programming a computer to analyze the syntax of the texts -- is beyond the present ability of both programmers and computers. Each word of the texts (63,000 words) was individually classified, the word-class translated into its numerical equivalent, each sentence thus being reduced to a series of significant numbers." (Milic, 1967, p. 143)

This numeric coding scheme was an adaptation of that proposed by the linguist Charles Fries (1952). In the version used by Milic there were 24 word classes, represented as 2-digit numbers. For instance, nouns were coded 01, auxiliary verbs 21, prepositions 51, and so forth.

After comparing the eight Swift texts with the eight control texts on a number of different attributes -- such as adjective-verb ratio, proportion of conjunctions etc. -- Milic selected three measures as reliable discriminators. On these he based what he called a "Swift profile", and claimed that "if it should be desired to test the attribution of a work to Swift, the matter may be quickly verified by reference to the three points of the profile". Using this method he concluded that *Letter of Advice to a Young Poet* probably was written by Swift.

The three discriminators forming his profile were: Verbals (VB), Introductory Connectives (IC) and Different three-word patterns (D). VB was the proportion of verb forms such as infinitives and participles, but not including main verbs. IC was the proportion of connectives, such as `but' and `for', beginning sentences. D was a measure of diversity that counted the number of distinct three-word groups used in a passage of approximately 3500 words: Swift scored higher than the other four authors on this variable. Milic also found that Swift was inordinately fond of long lists, as in: "I am not in the least provoked at the sight of a lawyer, a pick-pocket, a colonel, a fool, a lord, a gamester, a politician, a whoremunger, a physician, an evidence, a suborner, an attorney, a traytor, or the like" (though this last finding emerged from his own reading, not from computer processing).

Milic was not entirely complimentary about the work of Mosteller and Wallace, claiming, with some justice, that it demanded "a very sophisticated statistical technique", and that his method of using word classes rather than word tokens detected more frequent patterns whose significance could be assessed more simply in smaller samples of text. However, his suspicion of over-complex statistics took him perhaps too far in the other direction. Although his results were quantitative, they were not backed by any form of statistical reasoning.

This emerged as a weakness when Koester (1971) tried to employ the three-discriminator profile on other authors (Arbuthnot and Wagstaffe) who worked with Swift for the Harley ministry during the last four years of Queen Anne's reign. In the first place, despite extensive correspondence with Milic, she was unable in good faith to repeat his syntactic coding very accurately. This led, for instance, to a non-negligible difference of over 6% in the D score of a passage coded both by Milic and herself (844 versus 895). Second, and crucially, the discriminators failed to separate Swift from his contemporaries: three of her four Wagstaffe samples and two of the four Arbuthnot samples fell within the Swiftean range.

2.2.2 The Syntax of a German Romantic

Wickmann (1976) trod the same path as Milic by relying more on grammatical than lexical patterns in authorship attribution, and took this approach a stage further in applying it to a puzzle from German literature.

> "In 1804 a novel was published in Leipzig called *Nachtwachen* under the pseudonym `Bonaventura'. Because this novel is a key work of early German romanticism, literary historians have been trying to discover the identity of Bonaventura ever since it was first published. Among possible candidates some are well-known writers such as Jean Paul, Fr Schelling, Cl. Brentano, E.T.A. Hoffmann, but there have also been a few less-known ones like a certain Fr G. Wetzel." (Wickmann, 1976)

To solve this problem Wickmann took samples from works of fiction by five candidate authors who were active around the beginning of the 19th century (Brentano, Hoffmann, Klingemann, Jean Paul and Wetzel), as well as from Bonaventura, and parsed them by hand. This entailed allocating each word to one of 18 grammatical categories. From this data an 18-by-18 matrix of grammatical transition frequencies, containing 324 elements, was obtained for each work, then (making a few relatively unobjectionable statistical assumptions) a divergence measure, based on Chi-squared but transformed so as to be expressed as a z-score, could be calculated for various pairs of authors.

The method worked well in comparing texts from the five known authors; that is, when a work by Brentano, for instance, was evaluated using the transition matrix derived from a work by Jean Paul, the resulting divergence score was significantly different from chance expectation. When comparing the five candidates with *Nachtwachen*, only Hoffmann gave a score that was not statistically significant. All four other candidates differed from Bonaventura at the $p<0.02$ level or beyond.

Wickmann concluded that while the hypothesis that Hoffmann was in fact Bonaventura could not be rejected, the other four candidates were almost definitely excluded from being the true author of *Nachtwachen*.

This represents a well-conducted study with a clear-cut result on a genuine literary problem, but it does not seem to have been followed by other studies using the same method. Doubtless this is very largely due to the difficulty of manually assigning syntactic codes to long blocks of text. However, since the days of Milic and Wickmann, there have been great changes in automatic syntactic analysis. Although error-free parsing of unrestricted natural language texts is still not possible, a number of interactive systems now exist that make manual grammatical coding much faster, such as micro-EYEBALL (Ross & Hunter, 1994) which is quite inexpensive. There are also several tagger programs which assign grammatical tags to words in free-form texts without needing any human intervention, such as AUTASYS (Fang & Nelson, 1994) or the ENGCG system of Voutilainen et al. (1992). These attain high although not perfect accuracy in tagging English text, typically having more than 95% of their first-choice assignments correct when checked against expert human encoding[3].

It may be therefore that the time is ripe for a revival, and further development, of this abandoned grammar-based approach to authorship attribution.

2.2.3 The Credo of a Turbulent Priest

An important, if controversial, figure in the field of stylometry is the Reverend Andrew Q. Morton. We have already alluded to his early work on the authorship of ancient Greek prose, particularly the epistles of Saint Paul (Morton, 1965). He has developed his methods since then and applied them to other authorship problems. Indeed he was one of the first to apply stylometric tests in a forensic context. Here we concentrate on expounding some of the techniques described in his book *Literary Detection* (Morton, 1978).

With regard to inferential sophistication Morton's work represents a giant step backwards from Mosteller and Wallace, but a step forward in some other respects: for instance, he has pioneered the use of a wider range of diagnostic patterns than most others in this field. According to Morton (1978):

> "A test of authorship can be defined. It is some habit which can be numerically expressed and statistically described for which the works of one author can be regarded as a single population, and all the individual works, or parts of such works, can be treated as samples of that population. This means that the differences within his works, or between his works, in respect of this habit, will be only random sampling differences. If the test is to be an effective one, then the differences between authors in respect of the habit will be large and statistically significant."

Morton uses six main types of authorship test:

[3] There would seem to be no reason why such tagging systems should not already be, or shortly become, available in other major languages, including German.

(1) rate of function-word usage;
(2) sentence-length distribution;
(3) position of words in sentences;
(4) relative frequency of collocations (such as `such as');
(5) "proportional pairs";
(6) cusum charts[4], not just for description of verbal habits but also for inferring authorship.

Item (1) on this list is not peculiar to Morton: as we have just seen, Mosteller and Wallace used such tests extensively, so we will skip further discussion of them here. Item (2) was a favourite among early workers in this field but has since been discredited. Herdan (1965) called it "perhaps the most unsuitable of all available tests", while Smith (1983) concluded that the information provided by sentence lengths "is not sufficiently strong to warrant the use of such measures as a stand-alone technique to discriminate between authors"; so we will ignore that also. Morton's mode of using cusum charts has also been largely discredited (see, for instance, Hilton & Holmes, 1993), so we neglect item (6) as well. Thus we will consider here only Morton's use of items (3), (4) and (5) as authorship tests.

All three types of test can be illustrated by an example relating to the novel *Sanditon*, discussed by Morton in chapter 16 of his 1978 book. This novel was begun by Jane Austen but interrupted by her death in July 1817. Much later it was finished by "an Other Lady", using Jane Austen's notes, and issued as a complete work. Needless to say, the Other Lady did her best to imitate Jane Austen's style.

To test whether the Other Lady could be distinguished from Jane Austen, Morton counted the occurrence of various verbal habits in five chapters of archetypal Jane Austen, taken from *Sense and Sensibility* and *Emma*, as well as two early and two late chapters of *Sanditon*, by Jane Austen and the Other Lady respectively. His figures are summarized in Table 2.5.

[4] Such charts are routinely used in statistical quality control. A brief introduction to cusum charts can be found in Hogg & Ledolter (1992), for example.

Table 2.5 -- Verbal Habits of Jane Austen and the Other Lady Compared.

Morton's Data on Sanditon:			Sanditon 1:			Sanditon 2:	
Habit ↓	Rate in Emma & SS	Freq.	Expected	Chi-sq.	Freq.	Expected	Chi-sq.
an / (a+an)	0.13	11 /112	14.875	1.01	29 /112	14.875	13.41
such a / a	0.09	8 /101	9.1	0.13	2/83	7.48	4.01
and I / I	0.05	12 /151	7.11	3.36	1 /154	7.25	5.39
on the / the	0.03	8 /229	7.2	0.09	17 /221	6.94	14.56
the F.W.S.	0.09	19 /229	20.31	0.08	8 /221	19.61	6.87
this / (this + that)	0.26	15 /52	13.68	0.13	15 /37	9.73	2.85
with / (with + without)	0.83	28 /38	31.39	0.37	43 /47	38.83	0.45
d.f. = 6			Sum =	5.17		Sum =	47.54

The first column in this table shows that seven different verbal habits were used. All three types are illustrated here. An example of a "proportional pair" (which Morton sometimes also calls a "proportionate pair") is `an / (a+an)': this is simply the number of times `an' is used divided by the frequency of `an' and `a' added together. From the second column it can be seen that Jane Austen's rate from the two known works is 0.13 on this measure. An example of a positional test is `the F.W.S.' which is just the number of times `the' is first word in a sentence divided by the total number of occurrences of `the'. This ratio is 0.09 for Jane Austen. Finally, the habit labelled `such a / a' is an example of a collocation test: it is proportion of occurrences of `a' that are immediately preceded by the word `such'. For Jane Austen this figure is 0.09 to two decimal places.

The next two blocks of three columns show how Morton makes use of such measures in assigning authorship. In Sanditon-1 (by Jane Austen) there are 11 occurrences of `an' out of 112 occurrences of `a' and `an' combined. At Jane Austen's rate of 0.13 the expectation would be 14.875: from these figures a Chi-squared value of 1.01 is computed. In Sandition-2 there are also 112 occurrences of `a' + `an' combined, but in this case 29 of them are `an', giving a Chi-squared contribution of 13.41. To assess statistical significance these contributions are added for each test. For Jane Austen this total comes to 5.17, well within chance expectation; but for Sanditon-2, by the Other

Lady, total Chi-squared comes to 47.54 which, with 6 degrees of freedom, is very highly significant ($p < 0.0001$). Thus Morton concludes that the Other Lady was unable to imitate Jane Austen well enough to fool a stylometer.

The advantage of collocations and proportionate pairs, an idea introduced by Ellegård (1962), as used in this way, is that they are to some extent self-calibrating. Morton does not measure the absolute frequency of the collocation `on the', for instance, but rather the rate at which `the' is preceded by `on'. (Note that the rate at which `on' is followed by `the' would be a different test.) The disadvantage is that, as in the example above, some of these ratios might be expected to be correlated (e.g. `an/(a+an)' and `such a / a') and so the propriety of treating them as independent variables following a Chi-squared distribution must be taken on trust.

Morton's work has attracted widespread publicity, as well as severe criticism for lack of mathematical rigour (e.g. Smith, 1985). Indeed, in preparing Table 2.5 from his figures, I found a couple of apparent arithmetical errors as well as a more serious lapse: he had included a Chi-squared contribution from another collocation test for the Other Lady but not for Jane Austen on the basis of the fact that the **observed** frequency in the Sandition-1 sample was less than 5 but in Sandition-2 was more. If used, this criterion should be applied to **expected** frequencies. (See, for example, Porkess (1988).) Morton's pronouncements therefore cannot be accepted at face value, but must be carefully checked.

Nevertheless I believe that Morton has made important contributions to the field, particularly by introducing and/or popularizing new authorship indicators. He is, for instance, one of the few stylometrists who have attempted, by using collocations and positional information, to take account of one of the primary facts of language -- its **serial** structure.

2.2.4 A Plunge into Troubled Waters

Whereas Wickmann's investigation (subsection 2.2.2) involved identifying the writer of an anonymous novel and Morton's study of *Sanditon* sought to find evidence of a change of authorship within a single work, the extensive study carried out by Kjetsaa and colleagues (Kjetsaa, 1979) illustrates another of the ways in which authorship disputes present themselves -- an accusation of plagiarism, in this case levelled at the Russian novel *And Quiet Flows the Don* (published in four volumes).

The stimulus for this investigation was a book published in Paris by an anonymous Soviet critic known only as D* (1974) which accused the 1965 Nobel laureate Mikhail Sholokhov of plagiarism. As the eminent novelist Alexander Solzhenitsyn, himself another Russian Nobel prize-winner for literature, gave wholehearted support to the main contention of D* in a preface, this revived a charge that had been hanging over Sholokhov since 1928 -- that he had plagiarized nearly all of volumes I & II and more than half of volumes III & IV from a Cossack author called Fyodor Kryukov, who died of typhus in 1920 while taking part in an uprising against the Red Army.

Kjetsaa and a group of Scandinavian scholars addressed this question by subjecting seven text samples of at least 12,500 words each to computational analysis. Two were from undisputed works by Kryukov, two more came from undisputed works by Sholokhov, and the other three were parts I, II and IV of the *Quiet Don* novel (QD). Many linguistic variables were measured, but the four that Kjetsaa reports on in his 1979 paper are:

- sentence length;
- word length;
- vocabulary profile;
- type-token ratio.

Although mean sentence length did not differ between Kryukov and Sholokhov, their distributions of words per sentence, grouped into intervals of five, differed markedly: Sholokhov had a much higher proportion of sentences in the 6-to-10-word range than Kryukov (as did the author of QD). To test whether this discrepancy could be due to chance, Kjetsaa used the Kolmogorov-Smirnov statistic on the cumulative sentence-length distributions. This revealed that while the difference between Sholokhov and the QD author was what would be expected from random sampling variation, that between Kryukov and the QD author was significant at the $p<0.01$ level.

Mean word length did not differentiate the two authors, but it was found that Kryukov used a much higher proportion of 1-letter words than Sholokhov. This was due first and foremost to Kryukov's penchant for `i', which means `and' in Russian. This conjunction accounted for 5.78% of Kryukov's text, 3.67% of Sholokhov's and only 2.99% of QD. So on this measure also, Sholokhov is closer to QD than Kryukov.

The vocabulary profile showed that Sholokhov, like the QD author, used a greater number of *hapax legomena* (once-used words) than Kryukov. Conversely, the proportion of text made up by the most frequent 20 words was higher in both Kryukov samples than in either Sholokhov sample or in any of the three books of QD. This indicated that Sholokhov, like the QD author, employs a richer vocabulary than Kryukov, a finding consistent with Kjetsaa's assertion that "everybody reading 'the Quiet Don' in the original is struck by the novel's rich vocabulary" (Kjetsaa, 1979).

This impression was further confirmed by computing the type-token ratios of the first 12,500 words in each of the seven samples (12,500 words being the length of the shortest text among them). Type-token ratio is a simple measure of vocabulary richness: it is computed as the number of distinct words (types) divided by the total number of words (tokens). Expressed as percentages, the type-token ratios obtained by Kjetsaa were, in order of decreasing richness:

QD-2	51.46%
QD-4	50.67%
QD-1	49.27%
Sh-2	48.83%
Sh-1	48.34%
Kr-2	46.90%
Kr-1	43.23%

Once again Sholokhov comes closer to QD than Kryukov.

Taking these findings in conjunction Kjetsaa concluded that it would be unreasonable to maintain that Kryukov wrote the bulk of QD. Indeed Oakman (1980) quotes Kjetsaa as stating, in an unpublished report on this matter, that "the language seems to reveal that he [Sholokhov] wrote his own work, in which case the charge of plagiarism is null and void".

However matters are perhaps a little less clear-cut than Kjetsaa would have us believe. He lays great stress on vocabulary richness, as measured by *hapax legomena* and type-token ratio, and on the fact that QD exhibits a richer vocabulary than Kryukov's text samples. But it also exhibits a richer vocabulary than either of Sholokhov's samples. Since Holmes & Forsyth (1995) have conjectured, from results obtained in a different context, that writings produced by joint or collaborative authorship regularly have a more varied vocabulary than that of either contributor individually, the comparative richness of QD's vocabulary may not be as strong a confirmation of Sholokhov's sole authorship as Kjetsaa supposes. Indeed if this conjecture is true in general then this is exactly what would be expected on the assumption that Sholokhov re-worked material from notebooks of Kryukov that had somehow come into his possession -- just as Solzhenitsyn alleged.

This case, therefore, may still not be closed; especially in view of the fact that sentence and word length, the other two variables pointing to Sholokhov's authorship, are nowadays regarded with suspicion as authorial indicators.

2.2.5 Relative Vocabulary Overlap

Louis Ule (1982) studied the works of the Elizabethan dramatist Christopher Marlowe. He used four different measures to investigate whether Marlowe might have written a play called *Woodstock* (not usually accepted in the Marlovian canon) and a poem of 175 words beginning "I walked along a stream for pureness rare".

Two of these four measures -- relative vocabulary overlap (RVO) and distance between rare words used only once in a given corpus -- were invented by Ule himself.

RVO is a truly relational measure: it cannot be applied to a single text but only as a measure of similarity between two texts (unlike, for instance, ratio of `on' to `upon', which is a quantity derived from a single text). RVO depends on two quantities, Absolute Vocabulary Overlap (AVO) and Expected Vocabulary Overlap (EVO). To compute AVO, an ordered word frequency list is needed from both texts. The AVO is then the sum of the lesser of the two frequencies (including zero) with which each word appears in the two word lists. The EVO is defined as the number of word tokens that the two texts would have in common if they had been composed by drawing words randomly, without replacement, from the putative author's whole corpus. Dividing AVO by EVO gives RVO. Ule showed how to compute EVO without multiple random simulations and pointed out that it depends only on the lengths of the two texts and the vocabulary profile of the author concerned. Moreover he claimed that RVO has two very desirable properties: (1) it is not much influenced by changing the putative author from whose vocabulary structure EVO is calculated; and (2) it is gratifyingly stable across a wide range of text sizes. It can thus be used to compare a pair of texts with very different lengths.

Using RVO he proposed a chronology of Marlowe's dramatic output, based on maximal similarity between successive works. He also cast doubt on the belief that Marlowe wrote the short poem of 175 lines mentioned above.

In addition, Ule argued on the basis of a high RVO score between *Woodstock* and other Marlovian plays that it should be accepted as being from the pen of Marlowe. However, he explained away a high RVO between Thomas Kyd's *Spanish Tragedy* and *Massacre at Paris*, nominally written by Marlowe, as due to the close personal association between the two authors, which seems rather like special pleading.

Ule's novel measures, RVO and rare-word-interval, deserve further study. The latter has been developed since 1982 (see next subsection) but RVO appears to have been ignored by subsequent workers. Possibly this is due to the fact that nobody has any idea of its expected sampling distribution. A study with the aim of establishing RVO's sampling behaviour would be a worthwhile contribution to stylometry.

2.2.6 A Model of Rare-Word Usage

Most authorship studies have concentrated on commonly used words, in the belief that they are less liable to variations in rate of usage as a writer moves from one topic to another than rare words, and therefore that they offer the best place to seek an author's stylometric "signature". Such frequently-used function words are called "non-contextual" by Mosteller and Wallace (1984). Morton (1978) puts the case in favour of non-contextual words as follows: "however useful and attractive the rare words may be, the study of them has not been fruitful".

Some researchers, however, have gone against this opinion, among them Efron & Thisted (1976). Following a lead suggested by Sir Ronald Fisher, they developed a mathematical model for estimating the number of unseen species based on the rate of discovery of known ones. They proposed that the same logic could be applied to the problem of how many different words William Shakespeare knew but never used, based on the rate at which once-used words (*hapax legomena*) appear in the 884,647 word tokens in Shakespeare's accepted writings.

When Taylor (1985) found an anonymous poem of 429 words beginning "Shall I die" in the Bodleian library and declared that it was by Shakespeare, Thisted & Efron (1987) developed their earlier proposal into the basis of a test for authorship. From their model they were able to estimate how often each of the different words in a new poem of any given length by Shakespeare would be expected to have occurred exactly n times in his previous writings. Specially interesting is the case where n=0, i.e. words never used before by Shakespeare (corresponding in the ecological model to newly discovered species). In a poem of 429 word tokens they estimated that 6.97 words would be of this type, i.e. never previously used. In the Taylor poem there were 9 such words: \`admirations', \`besots', \`exiles', \`inflection', \`joying', \`scanty', \`speck', \`tormentor' and \`explain'.

From this model Thisted and Efron devised three distinct goodness-of-fit tests: (1) a test based on counts of words having values of n from 0 to 99; (2) a test based on the special case of n=0; and (3) a test of a slope parameter in a log-linear model used to fit these counts. The kind of data used is illustrated in Table 2.6 below, giving the observed and expected frequencies of various ranges of n from 0 to 99 in the Taylor poem. For example the row for n=2 shows that a 429-word poem would be expected to have 3.33 words used just twice in Shakespeare's other works whereas this poem had 5 such words.

Table 2.6 -- Words in the Taylor Poem Categorized by Shakespearean Frequency.

	Counts in a Poem of 429 words:	
No. of Prior Occurrences in Shakespeare's total Vocabulary:	Expected:	Observed:
0	6.97	9
1	4.21	7
2	3.33	5
3-4	5.36	8
5-9	10.24	11
10-19	13.96	10
20-29	10.77	21
30-39	8.87	16
40-59	13.77	18
60-79	9.99	8
80-99	7.48	5

The three tests were then applied to four Shakespearean poems, to three poems by other contemporary authors (Donne, Jonson and Marlowe) and to the Taylor poem. The results obtained on these eight texts are summarized in Table 2.7 below. Here the entries are z-scores -- essentially giving standardized deviations from expectation on the basis of the Null Hypothesis that Shakespeare is the true author.

Table 2.7 -- Summary of Thisted & Efron's Test Results.

Z-score on:	Test 1	Test 2	Test 3
Ben Jonson, Elegy	0.67	0.37	2.08
Marlowe, 4 poems	2.57	0.01	-4.04
Donne, `the Exstasie'	0.20	2.90	-1.53
Taylor Poem ("Shall I die ....")	2.29	0.16	-0.83
Shakespeare:			
from Cymbeline	2.86	0.00	-0.47
from Midsummer Night's Dream	0.37	-1.64	-0.42
The Phoenix and the Turtle	3.13	2.08	-1.41
Sonnets 12-15	1.24	-0.39	-0.38

From these results, Thisted and Efron concluded that test number 3 is the most useful and therefore that "the Taylor poem appears consistent with the hypothesis of Shakespearean authorship".

The logic behind this procedure seems questionable, however. They tried three related authorship tests and discounted the two that failed to distinguish Shakespeare from non-Shakespearean contemporaries. On the basis of the third, they ascribed the Taylor poem to Shakespeare. But since tests 1 and 2 did not work as intended, the accuracy of their model of rare-word usage for authorship attribution must be called into question. Test 1 is the simplest and most direct application of that model, yet -- taking a z-score of plus or minus 2 as a reasonable threshold -- it would give the poems by Jonson and Donne to Shakespeare while excluding two genuine Shakespearean works (as well as the Taylor poem) from his canon.

In effect, Thisted and Efron have conducted a search through a space of possible tests, stopping on finding one that discriminates successfully on a sample of 7 cases. To validate this test, it would next be necessary to apply it to a **fresh** set of texts -- i.e. to follow the method of Mosteller and Wallace by having both screening and calibrating sets (preferably much bigger than the texts used here, which amount to a mere 2750 words in total). In short, more empirical study of the efficacy of this test is required than that provided by Thisted and Efron. Thus the z-score of -0.83 for the

Taylor poem on Test 3 hardly provides compelling evidence for or against Shakespearean authorship.

2.2.7 A Touchstone for the Bard

Another criticism of Thisted and Efron's procedure has been made by Elliott and Valenza (1991) who pointed out that most of the words of Shakespeare used to calculate parameter values for their rare-word model were from drama whereas the model was applied exclusively to verse. Elliott and Valenza adopted a completely different approach to exploring the vexed question of Shakespearean authorship.

Their method is based on transforming 500-word blocks of text into numeric vectors, each element of which is the rate of occurrence of a particular keyword scaled according to its baseline frequency in the corpus of Shakespeare's poetry. They used 52 keywords from among the more common, but not most common, words in Shakespeare's poems, for example: \`again', \`beauty', \`might' and \`yield'. The 52-element vectors thus obtained were then subjected to a mathematical manipulation known as the Karhunen-Loeve transform. Essentially this produces a profile of each 500-word block with respect to the eigenvalues of the covariance matrix of the 52 keyword frequencies. (This transformation of text into a histogram recalls Mendenhall's (1887) quest for "word spectra" representing "characteristic curves of composition".)

If the eigenvalues are treated as weights and placed in descending order, a measure analogous to centre-of-mass can be computed from each of these profiles. From this measure Elliott and Valenza derived a scoring function where lower scores represent consistency with the base corpus.

For their first application of this method, which they call <u>modal analysis</u>, they divided Shakespeare's poetry into ninety 500-word blocks and compared it with 1446 500-word blocks from no less than 25 other authors claimed at one time or another to have written Shakespeare (including not only well-known "claimants" like Bacon, the Earl of Oxford, Marlowe and Sir Walter Raleigh but also Queen Elizabeth I) as well as from seven "non-claimants".

Shakespeare's 90 blocks were tested one at a time against the other 89, giving a cross-validated mean for their scoring index of 56.23 with a standard deviation of 40. Then the scores were computed for the other 32 poets against the 90-block Shakespeare corpus, leading them to state that "this test alone, in our estimation, cuts down the list of plausible tested claimants from twenty-five to zero" (Elliott & Valenza, 1991). The three candidates with organized support groups, Marlowe, Oxford and Bacon, were particularly distant from Shakespeare on this test, with their mean scores separated from the Shakespearean mean in terms standard errors by 4.85, 18.37 and 20.45 respectively.

A further study subjected some controversial anonymous poems, including the Taylor poem (subsection 2.2.6), to the same test. Elliott and Valenza concluded that none was by Shakespeare but that *The Passionate Pilgrim* whose authenticity has been challenged by reputable scholars (e.g. Wells & Taylor, 1987) probably is Shakespearean.

Modal analysis therefore would appear to be a useful addition to the stylometer's armoury.

> "Note that one of the most desirable features of this test is that by definition it accommodates the multimodality of a given author in a way that frequency based tests cannot. Word frequencies can vary sharply by subject-matter and genre, but our experience suggests that an author's modal structure can be remarkably persistent." (Elliott & Valenza, 1991).

A drawback of this approach is its lack of transparency. Although modal weight profiles make nice diagrams, with inauthentic passages showing plainly visible spikes in the wrong places, there is no natural interpretation of what the numbers signify.

Perhaps to remedy this, Elliott & Valenza (1995) have more recently subjected work (both plays and poems) by no less than 37 "claimants" to a battery of 52 separate tests, with the result that "no claimant, and none of the apocryphal plays or poems, matched Shakespeare." Two plays and a poem from the generally accepted Shakespearean canon, *Titus Andronicus*, *Henry VI Part 3*, and *A Lover's Complaint* also failed to match the rest of Shakespeare's corpus. In this follow-up study they used their modal test as in 1991, but supplemented it with an eclectic collection of additional discriminators, based on syntax, vocabulary and versification -- including percentage of feminine endings, frequency of `It' as opening word of a sentence, rate of usage of `I'm', `no / (no + not)' (a Mortonian proportionate pair), and many others besides.

A problem with this approach is that "to get the 52 tests we did use, we had to try hundreds of tests which did not show a sharp enough distinction between Shakespeare and others" (Elliott & Valenza, 1995). This would seem to be a serious methodological flaw. To rectify it, it would be necessary to take a subset of well-authenticated plays by Shakespeare and compare them with a number of plays by authors working around the same time who have never been seriously proposed as "true authors" of Shakespeare (e.g. Francis Beaumont and George Chapman). This would serve to validate the tests used by winnowing out those that failed to distinguish between Shakespeare and non-Shakespearean contemporaries. Then these tests could be applied with some degree of confidence to the rest of Shakespeare's plays, to the Shakespearean "Apocrypha and Dubitanda" and to plays by the genuine claimants. (*Mutatis mutandis* for poetry.) However, it does not seem that this more cautious procedure was followed, and to that extent the findings of Elliott & Valenza are less conclusive than they ought to be.

2.2.8 The Book of Mormon

A study by Holmes (1992) represents a refinement of the techniques used by Kjetsaa (1979) (subsection 2.2.4) and thus a further step away from reliance on individual word frequencies. This was an investigation of the Mormon scriptures.

The Mormon church claims that its holy book *The Book of Mormon* was translated by Joseph Smith in 1827 from golden plates which he discovered in New York State, engraved in a language called Reformed Egyptian. The book claims to recount the history of a family of Jews who escaped from Jerusalem just before its destruction in 597 BC and sailed to North America. Ever since its publication, doubts have been expressed about its authenticity. Holmes decided to subject the matter to stylometric study, using a method based on five measures of vocabulary richness.

He took 14 extracts from *The Book of Mormon* of approximately 10,000 words each, taking care to edit out reported speech so that the words of each extract were only those of the supposed author of the book concerned (*The Book of Mormon*, like the Bible, being divided into books by a number of different prophets). In addition, he obtained three text samples from private diaries and correspondence written or dictated by Joseph Smith as well as three samples from the biblical book of Isaiah (King James Version). Another Mormon scripture, *The Book of Abraham*, which Smith claimed to have translated from a papyrus written by the Hebrew patriarch Abraham that came into his possession in 1835, was also included. Finally, three segments of about 10,000 words each from *The Doctrine and Covenants* were also included in this analysis. This is a standard work of Mormonism, a collection of divine revelations given to Joseph Smith concerning the establishment and regulation of the Church of Latter-Day Saints, published at intervals from 1833 onwards. The main point about this work is that nobody claims it to have been written by anyone other than Smith.

For all these texts, five different measures of vocabulary richness were computed. A hierarchical single-linkage cluster analysis was then performed on this set of 5-dimensional vectors, using a Euclidean distance metric. This divided the texts into three main clusters: Joseph Smith's personal writings formed one group; the three extracts from Isaiah formed another, and all the rest of these samples fell into a third grouping. If the various extracts from this latter group had really been composed by at least seven different people, then this would be very surprising. Moreover, it contradicts a group of Mormon scholars (Larsen et al., 1980) who reported finding linguistic evidence of multiple styles within the *Book of Mormon*.

Next, a principal-component analysis was carried out on the five-dimensional data and the texts plotted as points in the space defined by the first two principal components. On this basis the texts appeared to fall into four main groups: Joseph Smith's personal writings; the chapters from Isaiah; two Mormon books (Nephi 1 and Mormon 1) as an outlying cluster; and all the rest together. The presence of *The Book of Abraham*, supposedly written almost 1500 years earlier than *The Book of Mormon*, near the middle of this large cluster cast considerable doubt on the proposition that the Mormon scriptures represent the words of several different authors, as does the fact that different extracts nominally from the same prophet (e.g. Alma, Mormon, Moroni and Nephi) showed no tendency to cluster together more tightly among than between prophets.

Holmes concluded that *The Book of Mormon* sprang from the "prophetic voice" of Joseph Smith himself, as did his revelations in *The Book of Abraham*. He also noted that the results of the principal component analysis could be replicated with just two of the five variables used, Honoré's R and the proportion of twice-used words in the vocabulary (*dislegomena*). Since the former measure (Honoré, 1979) depends on the proportion of once-used words, this suggests that the proportions of words used exactly once or twice in a text can be used as sensitive indices of authorial style. (Both measures have been shown to be stable for a given author in texts of more than 1000 words by Sichel (1986).)

A problem with this study, from the point of view of text classification, is that the allocation of texts to groups given a plot on the two dimensions of vocabulary richness is done on a rather informal basis. To turn Holmes's method into an automatic classification technique, we would need more information about the distribution of Honoré's R and of *dislegomena* within and between a wide range of authors.

It might also be argued by a devout Mormon that the clustering of texts allegedly by many different prophets into a single group, labelled as Smith's "prophetic voice" by Holmes, is a translation effect, i.e. that Smith imposed his linguistic habits as translator on originally diverse sources. However, the inclusion of the *Doctrine and Covenants* robs this objection of much of its force. This work is not claimed to be a translation (except perhaps from the word of God?) yet it clusters with texts that are so claimed. Hard evidence concerning the relative contribution of translator versus originator to the stylistic structure of translated texts is at present very sparse; but it would surely be a very heavy-handed translator who wiped out all trace of the original author.

Overall, therefore, this is an ingenious study that extends the range of available stylistic indicators and underlines the importance of multivariate methods.

2.2.9 A Multivariate Approach to Authorship

The work of J.F. Burrows represents an even more thoroughly multivariate approach to authorship attribution than that of Holmes, above. Burrows's methods have several variants and he has used them for other purposes than authorship attribution, but we will only consider here, briefly, his demonstration (Burrows, 1992) that they were capable of distinguishing quite easily between the three Brontë sisters -- Anne, Charlotte and Emily.

First he found the 75 commonest words in a file of about 56,000 words of text, containing roughly equal amounts written by each of the trio. These texts (retrospective first-person fictional narratives in each case) were then divided into thirteen blocks of approximately 4000 words each, five by Anne and four each by Charlotte and Emily. Then the relative frequencies of the 75 commonest words were calculated, giving 13 separate 75-dimensional vectors of numbers. From this data, treated as a 13x75 matrix, eigenvectors were computed. The next step was to plot the 13 samples in the 2D space of the first two eigenvectors (equivalent, in practice, to the two main components of a principal-components analysis). The three sets of extracts quite clearly fell into three distinct groups, corresponding to the three different writers. Except for one of Charlotte's extracts, which was nearer to one of Emily's than to any other of her own, every point on this graph had as its nearest neighbour another point from the same author.

Such a clear discrimination among three authors of the same gender, related both by heredity and by upbringing and writing at about the same time in a single narrative form, serves as a very convincing demonstration of the power of multivariate methods in stylometry -- which Burrows has extended to other sets of authors and to differentiating other aspects of style, such as linguistic change over time. In contrast with the laborious search for special verbal indicators of much previous work (e.g. Morton, 1978; Mosteller & Wallace, 1984) this method has the great merit of letting the texts "speak for themselves". It is particularly interesting in this connection, as Burrows points out, that "colourless" largely functional words have so much to say about patterns of relationship among authors.

Recently Binongo (1994) has confirmed the usefulness of this approach by applying essentially the same method, using in his case the 36 commonest words, to distinguish between three Filipino authors who write mainly in English -- with considerable success.

Nevertheless, this work does share the shortcomings mentioned in the previous subsection, as far as automatic text classification is concerned: (1) the dimensions defined by the two most important eigenvectors have no obvious interpretation; and (2) the method of assigning author to extract is only implied. Would we assign an unknown point to the class of its nearest neighbour or to the class of the nearest centroid? Do we compute Euclidean or some other distance metric in this space? Should we treat both dimensions equally or weight them by importance in some fashion?

2.2.10 Neural Nets and Shakespeare Studies

In recent years there has been a resurgence of interest in neural computation, which typically means programming conventional Von Neumann computers to behave as if they contained networks of simple processing units (modelled loosely on biological neurons) each connected to its neighbours by links of varying strengths, known as <u>connection weights</u>. An essential feature of such "neural" nets is that they can be trained by using algorithms that alter the connection weights between nodes in response to feedback comparing the net's actual with its desired output. It has been found that such artificial neural nets exhibit "emergent properties" that enable some hard computational problems to be solved (Aleksander & Morton, 1990; Wasserman, 1993). A favoured area of application for such systems is pattern recognition.

Stylometric researchers who have attempted to harness this pattern-recognizing ability include Thirkell (1992), Kjell (1994) and Tweedie et al. (1994). As an example of this type of work, a study by Matthews & Merriam (1993) concerned with the plays of Shakespeare and another Elizabethan playwright, John Fletcher, is described here.

Matthews and Merriam used a neural architecture called the Multi-Layer Perceptron (see, for instance, Beale & Jackson, 1990) with five inputs, two hidden nodes and two output lines, which they trained using the Back-Propagation training scheme. They wanted the network to learn to distinguish between Shakespeare and Fletcher so they gave it 50 training instances from known works by each author. Since Multi-Layer Perceptrons work with numeric inputs, these training examples were not in text form but were numeric vectors containing five stylistic indicators and an ID code. They compared two different sets of five indicators, based on previous work by Horton (1987) and Merriam (1992).

The Merriam set of five indicators were the following ratios: `did/(did+do)', `no/T10', `no/(no+not)', `to the / to' and `upon/(on+upon)' -- where T10 is the total frequency of 10 common function words suggested as diagnostic of Shakespeare by Taylor (1987). (Four of these indicators are Mortonian, either proportionate pairs or collocations: see subsection 2.2.3.) The five discriminators based on Horton's work consisted of ratios calculated by dividing the total number of words in the samples by the number of occurrences of the five function words `are', `in', `no', `of', and `the'.

They trained their nets on 100 training examples, 50 from each author, where each example was a feature-vector computed from a 1000-word block of undisputed text. Then they validated both nets on other undisputed examples, using a score they called "Shakespearian Characteristic Measure" or SCM to classify each example unambiguously. SCM was computed by dividing the value on the network's Shakespeare output line (a number between zero and one) by the sum of the outputs on both the Shakespeare and Fletcher output lines. A score above 0.5 was taken as a sign of Shakespearean authorship.

On a validation set of eight plays by Shakespeare and two by Fletcher, both nets gave 10 correct answers out of 10, based on this SCM score. But when individual acts, i.e. samples based on smaller amounts of textual information, were presented to the nets, the Horton-based network performed better than the Merriam-based network, with 85% correct classifications as opposed to 65% correct.

When used on plays of mixed and/or uncertain authorship, the Horton network allocated *Double Falsehood* and *The London Prodigal* to Fletcher and gave *Henry VIII* and *Two Noble Kinsmen* to Shakespeare, although when individual acts were examined it gave results suggestive of collaboration between these two authors in *Henry VIII*. Its verdict was that Acts I, IV and V were by Shakespeare while Acts II and III were by Fletcher. This agrees with majority opinion among literary scholars, derived from other sources of evidence.

According to Wilson (1993), Shakespeare suffered from "scrivener's palsy" by the time that *Henry VIII* was written and had given up play-writing for that reason. He had to be dragged back from retirement in Stratford to help with one last play because his theatrical associates had been persuaded to accept a commission to present a royal command performance to celebrate the wedding of King James's daughter, princess Elizabeth, in February 1613, and -- with the incomparable bard no longer part of their team -- were in danger of missing their deadline. Since the hand that, according to Ben Jonson, had scarce blotted a line, could by then hardly wield a quill, the up-and-coming dramatist John Fletcher was also drafted in to help complete the play on time. In the circumstances, it would have made sense to have allotted the opening scenes and finale to the old master of stagecraft, while assigning the young tyro to fill in the middle (perhaps also acting as amanuensis even on the Shakespearean portions). Thus the historical evidence, such as it is, fits in with what the neural nets found.

Matthews and Merriam broke no new ground as far as neural computing is concerned (Multi-Layer Perceptrons trained by back-propagation being quite oldfashioned in that field) but they did show that artificial neural nets could be applied successfully to a stylometric problem. They also found that, in this case at least, a set of indicators based on simple function-word counts were better discriminators than a similar set consisting of Morton-style collocations and proportional pairs. Undoubtedly this work will inspire others to follow them into "neural" stylometry.

A criticism that can be made of this work, however, is that their trained nets made mistakes on fairly large samples (entire acts of undisputed plays) even with only two candidate authors. Another criticism applies not just to their work but to all such systems. The "knowledge" gained by a neural network during training is inscrutable: it consists of a matrix of connection weights. If we simply want a black-box classifier, this may not matter; but if we want to understand what stylistic differences the system has discovered then we will be disappointed. Attempts to "de-compile" the knowledge implicit in a connection-weight matrix have been made (e.g. Sejnowski & Rosenberg, 1987) and research in this area continues, but the effort involved is typically an order of magnitude more than that of setting up and training the network in the first place. (For this reason, among others, Holmes & Forsyth (1995) have argued that artificial neural nets are not the most appropriate from of machine learning to apply in the stylistic arena.)

Another problem is that the SCM has no statistical underpinnings. Without further work it is impossible to know what evidential weight should be given to an SCM score of 0.77, for instance. It certainly isn't a probability, and is best regarded just as an index. (This contrasts with the scoring function of Elliott & Valenza (subsection 2.2.7) where an attempt to estimate its sampling distribution was made.)

2.2.11 Do Authors Have Semantic Signatures?

None of the studies reviewed thus far have tried to find semantic indicators of authorship. Indeed the concentration on "non-contextual" function words, by Mosteller & Wallace (1984) and Morton (1978) as well as others, is predicated on an assumption that the meaning of a passage of text gets in the way as far as determining authorship is concerned. A recent study by Martindale & McKenzie (1995), however, challenges this assumption.

Martindale and McKenzie (1995) returned to the *Federalist Papers* as a test case. They took it for granted that Madison wrote all 12 `disputed' papers and looked for alternative ways of establishing the same conclusion. They tried several methods, of which the most interesting was based on reviving the idea of content analysis, which was popular among social psychologists in the 1960s but which gave way to more formal methods of natural language processing as the field of computational linguistics grew to prominence. As they point out:

> "Although content analysis should be useful in author attribution, it has seldom been used. It is not that it has previously been tried with no success. It has simply not often been tried." (Martindale & McKenzie, 1995)

In one of their analyses they used a program, COUNT, which is essentially an updated reimplementation of the General Inquirer of Stone et al. (1966). This puts words into one of 55 mutually exclusive semantic categories. Ignoring function words, it usually categorizes about 80 to 90 percent of the words in a text. It uses a kind of thesaurus called the Harvard Psychosocial Dictionary (Mark III) which includes categories such as those in Table 2.8

Table 2.8 -- Examples of some Psychosocial Categories.

Semantic Category	Sample Words in that Category
Male Role	boy, brother
Thought Form	basic, contrast
Urge	eager, incentive
Ought	duty, ought, proper
Move	pull, run

Analysis of the relative frequencies produced by this program suggested that Hamilton used the categories Large Group, Bad, Urge and Ought significantly more than Madison, while Madison used words falling under the headings Quantity and Thought Form more frequently than Hamilton.

This implied that a discrimination rule could be based on such semantic information. Accordingly, the relative frequencies of these 55 categories were used to give numeric feature-vectors for all papers by Hamilton and Madison as well as the disputed papers. Then, having reduced the dimensionality of the data from 55 dimensions to five by Multidimensional Scaling, a linear discriminant function was formed on the papers of known authorship. This function, when applied to the disputed papers, classified 9 as by Madison, misclassifying 3 (numbers 50, 52 and 54) as Hamilton's. In addition, a type of neural-network classifier called the Probabilistic Neural network, or PNN (Specht & Shapiro, 1991), was trained on the same data. When this was used with the disputed papers it gave 10 to Madison and 2 (numbers 52 and 54) to Hamilton.

Thus the neural classifier appeared marginally superior to the linear discriminant -- provided we accept Mosteller and Wallace's verdict on the disputed Federalist papers. In any case, it would seem from this study that the practice of ignoring semantic information almost entirely in authorship investigations may be misguided. After all, this must rank as a hard problem for content-based discrimination, since both Hamilton and Madison were writing on essentially the same subject. As Martindale and McKenzie note, the semantic discrimination rules had no trouble with paper 55 (a problem case for Mosteller & Wallace's methods); thus content-based measures may have a valuable role to play alongside measures derived from function-word frequencies, since these two types of indicator tend to make mistakes under different conditions.

2.3 Other Relevant Research

Having looked in some depth at authorship attribution in section 2.2, this section turns first to other issues within the field of stylometry and then to other areas that are also relevant to the task of classifying texts automatically.

2.3.1 Stylometric Studies of Issues other than Authorship

Stylometry could (and to some extent does) concern itself with many issues other than authorship, of which the most relevant to the area of text classification are the problem of genre and the problem of chronology.

Genre refers to the different types of literary form -- ballad, drama, epic, essay, lyric, novel, short story and so on. There is no definitive list. Perhaps on account of this, genre has long been a neglected subfield within stylometry, although an early study by Radday & Shore (1976) found that the Hebrew definite article was a reliable marker distinguishing prose from poetry in the Bible, and Brainerd (1979) reported that pronoun usage appeared to co-vary with differences in genre.

Most authorship studies have treated genre as a confounding factor, liable to produce spurious results if not taken into account but not a subject of study in itself. However, the paper by Burrows (1992), cited in 2.2.9, contains an investigation into different literary forms used by Lord Byron and Sir Walter Scott, employing the same multivariate methods that Burrows used in distinguishing texts by the Bronte sisters. Relative frequencies of the sixty most common words in a combined corpus of prose, drama, verse and letters (both fictional and real) by both authors were obtained. Then principle components of the resultant matrix were computed, and points representing individual texts were plotted on a graph of the two most important components. Some interesting interlocking patterns of authorial differences and differences of genre emerged. A dramatic cluster could be seen clearly distinct from a cluster of letters: within both clusters texts by Byron were found towards the north-east while Scott's were located at the south-west. The procedure even separated fictional letters written by Scott in his novels from real ones. Although this study was purely exploratory, Burrows explained how, by selecting from the 60 or 75 commonest words a smaller subset of discriminatory ones, the method could be extended to become a classification procedure.

Another issue tackled by stylometers is the problem of chronology. Here the task is to arrange works by a single author in order of composition. This is not strictly a classification problem, though to reduce complexity it is sometimes turned into a problem of putting texts in categories, e.g. juvenile, mature, senescent.

The problem of dating Shakespeare's plays attracted the attention of one of the very first stylometers (Yardi, 1946). Yardi gathered data on the number of full lines, split lines, lines with redundant final syllables, lines with irregular scansion and other measures from the blank verse of Shakespeare's plays. He then divided thirty of the plays with relatively certain dates of first production into 21 successive groups of about one year each. Using this as a y-value he calculated a polynomial regression on the textual and metrical variables for each play. This enabled him to give estimated dates for each play and for other plays, whose date of composition was less certain. He noted a general tendency for Shakespeare to break away from strict formal rhythm as he grew older. A failing of this study is that there is not enough data to fit a function then validate that function on known examples before using it on genuinely uncertain cases.

Other researchers who have investigated the chronology of literary composition include Mansell (1974) who coded syntactic categories (by hand) in *The Old Man and the Sea* by Ernest Hemingway and in four other works by the same author written over a 20-year period. By examining a number of syntactic properties, he concluded that *The Old Man and the Sea* resembled earlier more than later Hemingway and hence that it was probably composed in 1935 or 1936, long before its publication in 1952.

Butler (1979) studied the poetic works of Sylvia Plath. Impressionistic observation had convinced him that Plath's poetry became less formal, although more complex in meaning, as she grew older. The variables he considered to test this observation were: relative frequency of end-stopped lines, total frequency of punctuation marks, relative frequency of formal (colon and semi-colon) versus informal (dash, question-mark and exclamation mark) punctuation symbols, sentence-length, word-length and type/token ratio. This latter variable is a crude measure of vocabulary richness (see also subsection 2.2.4). As expected he found a strong trend towards formally simpler language in the later poems: there were more end-stopped lines; sentences and words became shorter, and the ratio of informal to formal punctuation marks increased.

Butler appears to be alone in using punctuation as a stylistic indicator. Whether his findings about Plath would generalize to other poets is unclear. Certainly punctuation is more sensitive than most stylistic indicators to editorial intervention.

Brainerd (1980) returned to the question that Yardi (1946) had investigated, the chronology of Shakespeare's plays. He took 19 plays whose dates were not in doubt and measured several stylistic variables in them that exhibited strong covariation with date (frequency of certain word-stems, relative frequency of split lines of verse, mean word-length of verse lines and others). He then used these variables to predict the dates of certain plays for which the year of composition was in dispute. Once again, however, this was a study plagued by the difficulty of validating a predictive function on fresh but undisputed cases. As Brainerd himself puts it:

> "Ideally in a study such as this we should randomly divide the ... data into three parts, using the first part to find the best set of predictors, the second to obtain the best prediction equation involving the predictor-variables already obtained, and the third to obtain an independent measure of the error of prediction. However, with only 19 cases (plays), this procedure is not open to us, so we must proceed pragmatically and take the statistical consequences."

As a partial assessment of these statistical consequences, Brainerd conducted a leave-1-out cross-validation exercise.

> "I constructed 19 other such regression equations leaving out each time one of the 19 test set plays. They turned out to be highly unstable." (Brainerd, 1980)

The most marked difference occurred when *Two Gentlemen of Verona* was omitted. This must cast a shadow over the validity of the dates assigned using Brainerd's formula to plays whose date of composition is in real doubt.

In a modern replication of such a study it might be better to split each of the 19 well-dated plays into two or three parts (since a whole play is rather large as a single sample) thus giving, in effect, 38 or 57 samples and fulfilling the requirements for the 3-stage process outlined in the first quotation from Brainerd, above. (This option was not open to Brainerd in 1980 as he derived his figures from a previously printed concordance that treated each play as a unit.)

As already mentioned (subsection 2.2.5) Ule used relative vocabulary overlap (RVO), and other measures, in an attempt to date the plays of Christopher Marlowe (Ule, 1982). He wrote a program that tested all possible orderings of the seven plays (ignoring mirror-images) and chose the ordering with the lowest divergence score --

defined as the sum of the distances between adjacent pairs. Distance was computed in various ways: one that gave good results was just the inverse of RVO. Choosing a minimum-distance order corresponds to postulating that an author's style evolves only in small incremental changes. This is a reasonable postulate, but it cannot be considered proven: some authors might relapse into immature styles from time to time. Also, as Ule pointed out, to apply this exhaustive search to the 36 plays of Shakespeare would take billions of years. As far as I know, no one has used a modern heuristic search procedure, such as simulated annealing, to test this approach on larger sets.

Ule's work introduced a number of innovations but, at the risk of sounding repetitious, it must be said that he made no serious attempt to validate his method of dating. What is needed is a dating technique that consistently produces correct or near-correct orderings on a large assortment of texts whose dates of composition are well established but which have not been used in tuning the dating function, if one is used. I have found no report of such trials in my search through the stylometric literature.

It seems fair to observe, in closing this subsection, that workers in stylometry have tended to rush into real problems without testing their tools on unproblematic cases.

2.3.2 Text Discrimination as Pattern Recognition

As Holmes (1994) says:

> "Authorship attribution can be regarded as a special kind of pattern recognition where the pattern that is being searched for is the specific feature of the text that is thought to distinguish one author from another."

It might therefore be expected that authorship attribution, and text categorization in general, would be a normal branch of pattern recognition -- itself now subsumed within the field of machine learning (see: Forsyth, 1989; Weiss & Kulikowski, 1991). However, this is not the case. As seen in subsections 2.2.10 and 2.2.11, artificial neural nets (one species of trainable pattern recognizers) are starting to be used in authorship attribution; but, generally speaking, there is still very little cross-fertilization between the fields of machine learning and stylometry.

There seem to be more cogent reasons for this state of affairs than just the normal tendency towards ghettoization of academic subcultures. One such reason is that, until very recently, machine-learning systems have only been able to deal with modest amounts of data, typically in the order of a few hundred training cases, whereas practical text classification problems may involve a vast corpus of text. This is beginning to change as methods of "data mining" in large databases are developed (Piatetsky-Shapiro & Frawley, 1991).

A second, more important, reason is that most machine-learning systems work with numeric feature vectors as their normal input representation. In many applications (e.g. classifying sonar signals) the data arrives in numeric form, so this is feature-vector format is convenient; but it is not well suited to representing textual information. It forces the user to interpose a pre-processing step between the raw data and the learning system. Very often the choice of what features to present to the system is the crucial determinant of success; and this is the work of the user not of the computer. The work of Matthews and Merriam (1993), described in 2.2.10, exemplifies this: the hard work was done by the investigators in their choice of stylistic indicators. The neural net "learnt" how to weigh these indicators: it did not learn what indicators to use. Much the same applies to their follow-up study of Shakespeare versus Marlowe (Merriam & Matthews, 1994).

Thus, until the 1990s, scarcely any machine learning was carried out on text data. This situation is starting to change. For example, Masand et al. (1992) have developed a system for classifying news stories, as has Goldberg (1995); while Ahonen et al. (1993) report a system that learned a small regular grammar capable of describing the structure of all but 82 entries in a Finnish dictionary containing 15970 entries. A related study by Zhu & Shadbolt (1994) describes the development of a partial parser for extracting relational information from an encyclopaedic dictionary, as a prelude to building a knowledge base. Although in this case the objective was not to induce a rule for classifying unseen texts thereafter, it does show that that the fields of machine learning and text processing are no longer in watertight compartments.

As an exemplar of this new trend, a study by Apté et al. (1993) is outlined below.

The goal of this research was to create a system that would learn to classify documents, such as news-wire stories or scientific abstracts, into subject categories. At present the assignment of subject codes is done by people. Huge amounts of text are involved, so the process is expensive. Rule-based systems have been reported that classify documents about as well as humans (e.g. Hayes & Weinstein, 1991; Jacobs, 1993) but they have to be developed anew for each topic area, so are also costly. A system that discovered for itself how to assign subject codes to documents would be very valuable. This is a non-trivial problem partly due to size: the system would have to deal with hundreds of thousands of documents, using tens of thousands of potential attributes and dozens of category codes. Furthermore, a document may well belong to more than one category.

The work of Apté et al. (1993) on this problem consisted of three stages:

(1) a pre-processing step in which attributes used to describe text were selected;

(2) an induction step in which rule sets that would distinguish between classes were found;

(3) an evaluation step in which the rule set was pruned to minimize its expected classification error.

The initial task was to find a set of attributes. Apté et al. wrote a program that scanned texts for words and word pairs that occurred at least five times. In contrast with the studies described in section 2.2, they excluded high-frequency function words, so their indicators were contextual words and pairs of such words. The program found between 1000 and 10,000 indicators, depending on topic.

The next step involved applying the SWAP-1 rule-induction algorithm (Indurkhya & Weiss, 1991) to a large training set of texts using the attributes identified in step 1, above. SWAP-1 uses a heuristic search to find the conjunctive rule (involving logical AND between attributes) that covers as many instances of one class as possible but no counter-examples. Once found this rule is added to the rule-set and the cases it covers are removed from the training data. If cases still remain, the process is repeated: another such rule is induced and the cases it covers are removed. This goes on till the example set is empty. To cope with more than two categories, the program is simply run several times, each time learning a rule-set that distinguishes between a particular category and all the other categories treated as negative instances.

The sort of attributes used and rules produced are illustrated below from a test where the system learned to distinguish between news stories about (American) football and non-football stories.

kicker	=>	football article
injure reserve	=>	football article
award & player	=>	football article

Line 1 shows a simple indicator, a content word; line 2 shows what stylometers would call a collocation; and line 3 shows a logical conjuction of indicators, which, unlike a collocation, need not appear sequentially.

The final step was to prune these rules down to a compact covering set on the basis of performance on a set of validation texts. In this respect working with large data sets (one of their trials was on a collection of over 10,000 Reuters news stories) was a blessing: it allowed them the luxury of using large enough samples in both training and validation phases to derive statistically reliable estimates of likely error rates on fresh cases. This meant that they could purge from the rule-set rules that only covered anomalies in the training data.

Results from a study of stories taken from the United Press International news-wire service are tabulated below.

Table 2.9 -- Recall & Precision on News-Wire Stories.

Subject	Recall	Precision
Air Transportation	57%	89%
Football	87%	95%
Hockey	84%	91%
Mergers & Acquisitions	39%	71%

These results were obtained on a sample of over a thousand unseen cases. The terms recall and precision are used as in the information-retrieval literature: recall is the percentage of cases in database of the desired category that are selected by the rules; precision is the percentage of cases selected by the rules that do in fact belong to the category sought.

It can be seen that this system excelled at picking out sporting stories but that performance on what might be thought the most financially useful task (finding articles about mergers and acquistitions) was poor. Apté et al. looked at some of these stories and conclued that the human coders had made many classification errors on them! (This interpretation may well be right, given the somewhat nebulous nature of such a category heading.)

Although not above reproach, this study does indicate that the time has come when machine-learning techniques can usefully be applied to a range of practical text-classification tasks. In particular, it suggests: (1) that the laborious hunt for textual attributes or indicators could be done by machine; (2) that what a learning system has learned can be represented in a compact and reasonably intelligible form (e.g. as a "rule base" rather than as a mysterious matrix of connection-weights); and (3) that there is no excuse for neglecting the essential validation step in developing ways of discriminating between types of text.

Other recent studies which tend to support this conclusion include work on classifying inter-bank financial telexes (Sahin & Sawyer, 1989; Goodman, 1990) using both rule-based and case-based systems; as well as work by Lehnert et al. (1995) who report on an inductive system that learnt to distinguish two types of "encounter notes" concerning asthmatic patients (written by physicians and held on a computerized medical database) with some success.

2.4 Review

This section is an attempt to draw together the various strands described in previous sections in order to portray the current state of the art in text classification and to suggest ways of making further progress.

2.4.1 A Perspective

A major premiss of this thesis is that text classification is usefully regarded as a type of pattern recognition and therefore that techniques from the field of machine learning can be applied to authorship attribution and other text-classification tasks. From this point of view, the fact that there are some recent signs of convergence (outlined above) between the previously separate areas of stylometry and machine learning is a development to be welcomed. The present work is an attempt to take that convergence a step further.

This view helps to provide a framework for organizing the diversity of studies described in this chapter which (although some important work has been left out in the interests of brevity) still present a wide range of techniques applied in a variety of domains. From this standpoint, the goal of text classification is not so much to assign a small number of texts to their correct category but to develop a rule or procedure for doing so, i.e. to produce a classifier. This will typically involve several stages. Thus a generic text-classification task falls naturally into five main phases:

(1) Data collection, possibly including pre-processing;
(2) Feature selection;
(3) Induction of a rule based on selected features;
(4) Validation of the rule or rule-set;
(5) Application of the rule.

(Apté et al. (1993) describe only the middle three of these stages, as noted in subsection 2.3.2, perhaps considering the initial and final stages too obvious to mention.)

The sort of activity conducted in each of these stages can be illustrated with reference to one of the earliest (Mosteller & Wallace, 1984) and one of the most recent (Matthews & Merriam, 1993) of the stylometric studies described in this chapter.

Stage 1 for Mosteller and Wallace involved the collection and encoding of over 200,000 words of known authorship by their two candidate authors, Hamilton and Madison. Matthews and Merriam were able to take their samples of Fletcher and Shakespeare from existing computer-based text archives. In the early 1960s, punching text onto cards was possibly the most arduous part of the whole enterprise; but these days there is almost an embarrassment of available text and the problem may soon become one of selection. This part of the exercise may seem just an intellectually trivial chore, but it should be noted that some decisions made at this stage (such as conversion to upper or lower case, insertion (or removal) of "mark-up" codes &/or diacritics, "lemmatization" or reduction of inflected words to root forms, and so on) can have profound and unforeseen repercussions on later stages.

Stage 2 is the search for what Mosteller and Wallace call "markers" and what Matthews and Merriam call "discriminators". The latter were content to employ two sets of five discriminators, proposed in previous studies, while Mosteller and Wallace made a more extensive trawl through many hundreds of potential marker words; but in neither case was the vast number of potentially discriminating patterns searched exhaustively. Most studies, including the two here mentioned, have used words as indicators or else numeric measures derived from word counts; but if substrings smaller than words (e.g. prefixes such as `un-' or suffixes like `-ed' or `-ation') were taken into account as well as collocations (e.g. `of the' or `more or less'), perhaps with "wild cards" (e.g. `in th* house*' to cover `in this house', `in the houses' etc.), it can be seen that there is a vast space of potential indicators which no previous investigation has explored more than superficially.

The next stage (number 3 in the list) involves the collation of individual indicators into some sort of coordinated structure which a machine-learning researcher would term a <u>rule base</u> or <u>knowledge base</u>. Both Mosteller & Wallace and Matthews & Merriam could be said to have created rule bases -- one consisting of a table of 30 function words each with six associated distributional parameters, the other consisting of the weight-matrix in a trained neural network -- but in neither case was the structure of the rule base a primary concern. It is here that the contrast between practice in stylometry and machine learning is most striking, because it is thought very important by most workers in the latter field (even some in neural computing) that any "knowledge" developed by a learning system should be intelligible and, in a certain sense, open to inspection.

Stage four (Validation) was called "calibration" by Mosteller and Wallace. They carried it out with extreme thoroughness. They were fully aware of the possibility that a list of markers selected for their discriminating ability on one sample of text might prove much less effective on another sample. With this in mind they estimated the efficacy of their list of marker words plus parameters on a calibration set which had not been used in selection of the markers or in estimating their parameters.

Matthews and Merriam also tested their nets on plays of known authorship that were not used in training those nets, to estimate their likely error rate, before letting them loose on disputed data. However, many other researchers have dived straight into problematic cases, such as the integrity of the Pauline epistles, without validating their classifiers. A widespread recognition that this stage is indispensable would be an advance.

The final stage (number five) is the application of the knowledge gained, in whatever form it takes, to particular texts. For Mosteller and Wallace these were the disputed *Federalist* papers. In the case of Matthews and Merriam they were four plays of uncertain authorship. There is nothing intrinsically complex about this part of the process; but there is no general agreement about how its results should be presented. Mosteller & Wallace were able, because they had adopted a Bayesian framework, to quote posterior odds on Hamilton's (or Madison's) authorship of each paper. Matthews and Merriam gave scores whose evidential import is unclear. Others, such as Burrows (1992), have simply presented graphs which look suggestive but which, as he points out, do not remove the responsibility of judgement from the reader. In short, most classifications come with some degree of uncertainty and the best way of handling this uncertainty is not a solved problem.

The fact that this five-stage framework fits both a relatively early and a relatively recent stylometric study without too much distortion tends to confirm the value of looking at authorship attribution, and other types of text categorization, from a machine-learning perspective.

2.4.2 Digression

At this point the reader may well ask: if text classification is a branch of machine learning and if its main object is to discover rule sets for putting texts into categories, why not just use a grammar-induction system on samples of each type of text to be distinguished and let it learn the idiolects of the corpora concerned; then assign fresh texts according to which grammar (`idiogrammar'?) is best able to parse them? It is worth digressing a moment to reply to this question.

The short answer is that it is too hard. A longer answer is that although nowadays hand-crafted parsers are capable of assigning correct syntactic tags to over 96% of words in large volumes of unrestricted English prose (see, for instance, Heikkila & Voutilainen, 1993) and could therefore be useful in a modern replication of a study like that of Milic (1967) on Swift's grammatical habits, they would not be of much help if the texts to be compared were, say, in ancient Assyrian or were transcripts of utterances by patients with two different types of aphasia -- or even DNA sequences encoded alphabetically. In none of such cases would a grammar for English, however comprehensive, be very helpful. In other words, many practical text-classification jobs do not involve grammatical English prose, even within the English-speaking world.

To address this wider range of problems by syntactic analysis would require a system able to learn a grammar anew for each fresh classification task. Here the current state of the art is far less advanced (see, for example, Wolff, 1991). Systems have been devised that can learn artificial grammars on restricted alphabets such as binary strings (e.g. Porat & Feldman, 1991) or subsets of English such as auxiliary verbal phrases (e.g. Berwick & Pilato, 1987); but a general-purpose grammatical induction engine remains a dream for the 21st century. Meanwhile, many kinds of text classification are perfectly possible without such a device: I can distinguish written (or indeed spoken) Chinese from Japanese knowing less than 20 words in both languages combined.

2.4.3 Desirable Developments

The five-step framework (subsection 2.4.1) does give some general guidance on how to go about a text-classification task, but it leaves unanswered an important question: what specific novel developments are desirable in order to make progress in this area?

A simple, and perhaps apposite, way to answer this question is to imagine that Mosteller and Wallace were just beginning their investigation today, and consider what extensions and/or improvements they might be tempted to make, given modern computing facilities and techniques. (This is predicated on granting their work exemplary status and admitting that it is still surprisingly up to date, despite the passage of three decades; but enough has been said already to demonstrate that in many respects, especially with regard to statistical sophistication, there has been some backsliding by subsequent researchers.)

As far as data collection is concerned, Mosteller and Wallace would find the situation much easier if they were embarking on their project today. (For example, an electronic version of the entire text of the *Federalist Papers* is in the public domain: it can be down-loaded at no cost from Project Gutenberg.) In these circumstances they would no doubt conduct an even more impressive study than they did. Some of the improvements and extensions that might be made are listed below, organized under the headings of the five-step framework outlined earlier.

(1) Data Collection:
   - ➢Deal with more than 2 categories.

(2) Feature Selection:
   - ➢Automate the hunt for markers;

- ➢Use markers other than words, both below the word level (e.g. suffixes) and above (e.g. collocations).

(3) Rule Induction:

- ➢Try a range of modern learning algorithms;
- ➢Place more emphasis on **representing** the discriminatory knowledge gained during the learning process.

(4) Validation:

[On this point Mosteller and Wallace were more thorough than most of their successors; nevertheless improvements may still be possible.]

- ➢Attempt to find ways of efficiently incorporating cross-validation or related techniques, such as jack-knifing (Mosteller & Tukey, 1977) or boot-strapping (Efron, 1983), which, though computationally expensive promise more accurate error-rate estimation than simple division into training and test sets.

(5) Application:

- ➢Find ways of dealing with smaller chunks of text.

The first four proposed improvements should be self-explanatory, but two of them (seeking better ways of representing discriminatory knowledge and being able to make decisions on smaller extracts) deserve slightly more elaboration.

Representation has long been recognized as fundamental in knowledge engineering and machine learning (see, for instance, Cupit & Shadbolt, 1994). Indeed it is a key issue in all branches of computer science. For example, Brooks (1982) states that representation is the essence of programming, and writes:

> "Show me your flowcharts and conceal your tables, and I shall continue to be mystified. Show me your tables, and I won't usually need your flowcharts; they'll be obvious."

Yet this aspect of text classification has been virtually ignored by previous workers. This is illustrated by the fact that there is no name for the structure representing the classification rule or procedure (corresponding to knowledge base in the field of Expert Systems). Various terms have been used for the elements of which it is composed, such as "marker", "indicator", "discriminator" (although there is no consensus on this) but no term exists for the coordinated structure built from these elements. For the sake of having a handy label, the term stylogram is introduced here to refer to such things.

A stylogram is in some sense a model of the distinctive characteristics of a type of text. It is likely that paying more attention to the choice of stylogram would be a good idea.

It could be expected to have two beneficial consequences. Firstly, a focus on representations and their properties could be expected to lead to a wider range of structures being used -- today's somewhat narrow range of structures being almost certainly less than optimal for many of the tasks to which they are applied. Secondly, and perhaps more important, a greater emphasis on representational adequacy might lead to the development of stylistic structures that could be used for descriptive as well as inferential purposes. That is to say, considering the intelligibility of stylograms might well lead to the development of structures that could be read as compact summaries of what is distinctive about a certain kind of text, even when not being used to categorize such texts. This in turn should lead to greater insight into the properties of the texts being analyzed.

The question of size is perhaps a more down-to-earth issue, but no less important. An examination of previous studies reveals that they have tended to need rather large slabs of text to make a classification, as shown by Table 2.10.

Table 2.10 -- Size of Text Blocks Analyzed in a Selection of Stylometric Studies.

Researchers	Subject	Norm	Range
Smith (1985)	Elizabethan drama	20595	17965-24925
Ule (1982)	Marlowe's writings	13000	175-21106
Holmes (1992)	Mormon scriptures	10000	5715-12776
Butler (1979)	Sylvia Plath's poems	8000	6100-9340
Merriam (1989)	Federalist papers	6000	[unknown]
Burrows (1992)	Brontë sisters	4000	500-8000
Milic (1967)	Jonathan Swift	3500	3324-3777
Morton (1965)	Saint Paul's epistles	2600	286-7592
Mosteller & Wallace (1984)	Federalist papers	2000	906-3506
Ledger (1989)	Platonic dialogues	1000	1000-1000
Matthews & Merriam (1993)	Elizabethan drama	1000	[unknown]
Elliott & Valenza (1991)	Elizabethan poetry	500	425-500
Thisted & Efron (1987)	Shakespeare & others	400	234-495

This table collates information from several stylometric studies showing the length, in words, of text blocks that various researchers have attempted to categorize. The numbers in the column labelled Norm give the typical or recommended text size for the researcher(s) concerned, i.e. the size for which they have confidence in their methods. The rows have been arranged in descending order of this norm. (Morton (1965) recommends taking blocks of about 200 sentences in length: the values in his row have been calculated by multiplying his sentence lengths by 13, since he quotes that figure as an average sentence-length in biblical Greek.)

The Range column gives the size, again in words, of the smallest and largest text analyzed by the worker(s) concerned.

It will be seen that there is wide variation. None the less, the size of text block that previous researchers have felt able to categorize is typically quite large, the median in the Norm column being 3500 words. The smallest text block ever given an attribution (so far as I can ascertain) was a 175-word poem analyzed by Louis Ule (1982) who also analyzed the next-smallest, of 192 words (as well as some of the biggest). It is exceedingly rare for anyone to attempt to classify a segment of text less than 250 words in length.

When it is considered that the median length of Shakespeare's sonnets is 112 words, this is a definite limitation. It would be a real advance to have a method capable of reliably classifying texts less than 200 words in length, preferably as short as 100 words. Whether this is possible is an open research question; but it is certainly a goal worth aiming at.

2.4.4 Prospect

Having surveyed some previous research in text classification and indicated some potential lines of advance, we turn next to a study using a novel text-classification method that does not, in fact, fulfil all the desiderata proposed in the preceding subsection -- but which does, nevertheless, serve as a starting point for a connected series of studies aimed at making progress towards fulfilling them.

Chapter 3

TEXT CLASSIFICATION BY CROSS-CODEBOOK COMPRESSION

> "The statistical methods used in the great majority of works on computational stylistics take no account of the serial nature of language; a sentence-length distribution, or a word frequency list, does not preserve the order of words or sentences." -- David Holmes (1985).

This chapter describes a novel approach to text classification and evaluates the results of applying it to some sample text sets. The method is based on ideas drawn from the field of data compression.

3.1 Why Data Compression?

It may seem strange to begin a project dedicated to bringing machine-learning techniques into stylometry with a method based on data compression, but in fact there is much common ground between these two areas, as has been noted by previous authors (e.g. Wolff, 1991).

Generally speaking, a learning system, if successful, comes up with a **compact** description of (aspects of) a collection of training instances. Such a description may be said to have extracted most of the redundancy in the training data. Conversely, a data-compression program may be said to have a **model** of the data source it is dealing with. The importance of modelling in data compression has been stressed by Witten et al. (1987), among others. Moreover, if a compression program uses an adaptive algorithm it will have refined or tuned that model with experience. Hence these two subfields with apparently different aims and subject-matters are in fact intimately related. Indeed Cheeseman (1990) has proposed the principle of Minimum Message Length as a fundamental principle of machine learning, building on earlier work by Wallace & Boulton (1968), Wolff (1976) and Rissanen (1987).

Going further back, it was Claude Shannon, one of the founders of Information Theory, who first pointed out (1951) the essential link between predictability and compressibility: if we have a good model of a data source then we can predict its behaviour well; to the extent that we can predict the behaviour of a data stream, we are able to devise a coding scheme that will encode it more economically than its original form.

This is the theoretical grounding for the present work, which was motivated by the observation that in practice some data-compression techniques can become `over-specialized'. That is, a compression technique that is highly tuned for one type of source (for instance, English prose) may perform poorly when applied to other types of data (for instance, C programs). In extreme cases, the `compressed' form of the data may even be larger than the original file.

When this happens, it means that the implicit model of the data source used in the compression program is so well adapted to the idiosyncrasies of that particular source that it no longer fits other sources of data very well. From a data-compression viewpoint, this is an unwanted side effect; but from the present perspective it suggests that data compression could serve as a method of identification, with degree of compressibility used as an index of compatibility between model and data. In other words, the programs described in this chapter represent an attempt to turn what is usually seen as an undesirable side-effect into a main effect.

3.2 The Algorithm Used

There are many different compression techniques, suited to different kinds of data -- image, voice, numbers, text etc. However they all share the same basic objective of economizing on data storage and/or transmission by exploiting redundancy inherent in the `raw' data (Arps, 1979).

Textual data, normally represented as streams of byte-encoded characters, is of major concern in many computing applications and text compression has been intensively studied. Storer (1988) gives a comprehensive discussion. In the present context, data compression as such is not the primary objective and so a particularly simple algorithm was chosen as the basis for this investigation. It is an adaptation of the pairwise chunking algorithm described by Dawkins (1976) which was in turn a minor modification of the MK10 program invented by Wolff (1975). It is also closely related to the technique termed diatomic encoding by Held & Marshall (1991).

Dawkins is a zoologist; so the first application of his program was in a zoological investigation, an attempt to reveal the sequential structure of acts observed in the grooming behaviour of blowflies.

Firstly, the actions of the flies were encoded as sequences of integers, signifying a succession of acts in their order of occurrence. (They could easily have been encoded as characters.) Next the program scanned through sequences of this kind counting frequencies of doublets. The program printed the most common doublet and then went back over the data series, replacing each occurrence of the doublet by a single symbol representing that pair of acts. This process was repeated cyclically on the data series until a stopping condition was satisfied.

An advantage of this algorithm is that it can build up quite long chains -- if they occur in the data -- thus identifying sequential dependencies of quite high order (in a Markovian sense) without demanding excessive computational resources. In particular, it does not need the huge, but sparsely filled, multi-dimensional matrices that would be required by a simple-minded approach to analyzing transitional probabilities spanning more than a very few items at a time.

A program embodying this algorithm (called DIGS, for digrams) was written in C. DIGS takes an ASCII text file and produces a codebook that can later be used in compressing that file, or others. It makes the assumption that it will be dealing with sequences of 7-bit ASCII codes and hence that character codes from ASCII 128 and upwards are free for reassignment. (This assumption means that it would require modification to handle texts in French, German or other languages using the full 8-bit ASCII code, or binary files.)

DIGS scans an input text looking for the most frequent pair of characters and at the end of each scan replaces each occurrence of that pair by a newly allocated digram code, starting with ASCII 128. It repeats this process till the most common pair occurs less than eight times or the requested number of pairings have been made. Its main output is a codebook -- a list of the codes allocated to each digram -- which can be used to encode (and compress) other text files. Compression is achieved by representing common substrings with single bytes.

We are not specially interested in blowflies, so as illustration an extract from an example codebook, derived by DIGS from the Dylan Thomas poem *Altarwise by Owl-Light* is shown below.

Table 3.1 -- Selected Doublet Frequencies in a Poem by Dylan Thomas.

Code	Doublet	Expansion	Frequencies
128	101 32	e_	200, 73
129	116 104	th	188, 20
130	32 129	_th	137, 25
131	130 128	_the_	112, 96
132	115 32	s_	101, 54
133	100 32	d_	99, 25
134	105 110	in	88, 43
135	97 110	an	73, 31
....			
149	39 132	's	33, 33
....			
155	134 103	ing	29, 15
....			
157	135 133	and_	27, 27
....			
178	134 131	in_the_	16, 16

[Here the underline character ('_') has been used in place of the space character (ASCII 32) to make blanks more visible.]

In this table the first column shows the code number allocated to the digram listed under the heading Doublet, whose text expansion is given in the next column. Lastly, there are two numbers under the heading Frequencies. The first of these is the number of occurrences of the given digram found during the scanning process; the second is the number of occurrences of that particular code in the final, compressed, text. It is important to realize that these two numbers do not have to be equal, since a doublet may be subsumed into a longer substring as scanning proceeds. For instance, the second line shows that there are 188 occurrences of the pair `th' in this text, but by the time the scanning is finished only 20 occurrences of the specific code for `th' (number 129) are left. This is because occurrences that are part of longer substrings, such as `_the_', disappear when those substrings are replaced by their own codes, which are listed further down the table.

Another point illustrated by this example is the tendency of the method to pick out common morphemes, words and collocations such as `ing', `and' and `in the'. This happens naturally as pairs are repeatedly joined together. Reversal of this embedding process allows a compound code to be unpacked into its constituents. For instance, code number 131 (`_the_') results from coupling code number 130 (`_th') with code number 128 (`e_'), and thus can be decomposed as ((_(th))(e_)). Likewise code 157 (`and_') can be `parsed', in a sense, as ((an)(d_)), though whether this is linguistically meaningful is open to question. It was this property of the procedure that most interested its originators, Wolff and Dawkins, who were chiefly concerned with grammatical or quasi-grammatical analysis, although we will not pursue it further here.

In addition, programs PACK and EXPRESS were written to compress and expand text files using codebooks produced by DIGS, though this was more to check that the method worked than as a serious attempt to save space on disk. More relevant to the present purpose was a program called DIGTEST which could apply several codebooks to several text files and provide various statistics including, of course, their sizes after compression.

3.3 Some Results

The hypothesis guiding this work was that a codebook captures some of the sequential regularities characterizing the text from which it is made, and therefore that a text should be better compressed by a codebook derived from texts of the same type than by a `foreign' codebook developed from a different source.

To test this hypothesis, and thereby gain an idea of whether this insight could serve as the basis for a workable method of text categorization, the DIGS and DIGTEST programs were applied to an assortment of text files.

A pilot study showed that the method had little trouble in distinguishing between Bob Dylan and Dylan Thomas (not indeed a very severe test) so a slightly more stringent and somewhat more realistic task was devised as an initial trial of the system.

3.3.1 A Test of Authorship

Firstly a considerable amount of English poetry was gathered and saved on disc, much of it scanned and edited or typed directly by the present author. (This work is ongoing: it has involved assembling what begins to amount to an electronic anthology, of which more details are given in Appendix D.) Next the writings of the four authors for whom the largest text samples were available at the time were divided into two portions, on the basis of when they were written during each poet's career -- labelled for present purposes `early' and `late' work. These poets were Bob Dylan, Dylan Thomas, John Pudney and T.S. Eliot[5]. Then 80-item

[5] Bob Dylan's status as a major twentieth-century poet has been endorsed by Allen Ginsberg and Professor Christopher Ricks (Ricks, 1984), among others, and I am happy to concur.

codebooks were formed by DIGS on the four text files of early work (referred to below as BD.1, DT.1, JP.1 and TE.1) and used to compress the late text files (referred to as BD.2, DT.2, JP.2 and TE.2). In this, as in all the tests reported in this section, the files used were plain text files which were not pre-processed in any way except that: (1) upper case was folded to lower case throughout and (2) tabs were converted to (single) blanks.

Results are shown in Table 3.2.

Table 3.2 -- Results of Cross-Codebook Compression with Four Poets, Early versus Later Work.

Code Book → Text File ↓	BD.1 (65913)	DT.1 (43241)	JP.1 (25898)	TE.1 (42777)	Row Means ↓
BD.2 (42459)	25469 0.5998	26557 0.6255	25586 0.6026	26264 0.6186	0.61163
DT.2 (23771)	14323 0.6025	14017 0.5897	14175 0.5963	14094 0.5929	0.59535
JP.2 (29252)	18371 0.6280	18480 0.6318	17990 0.6150	18211 0.6226	0.62435
TE.2 (31907)	19576 0.6135	19221 0.6024	19068 0.5976	19060 0.5974	0.60273
Column Mean =	0.61095	0.61235	0.60288	0.60788	

In this table the upper figure in each cell of the last four rows/columns is the resultant file size in bytes, the lower figure is the ratio of the compressed file size to its original size. Thus, for instance, the size of JP.2 (late John Pudney) after compression by the code book generated from DT.1 (early Dylan Thomas) was 18480 bytes or 63.18% of its uncompressed size. Also the number of bytes in each file and in the files from which the codebooks were developed is given in brackets under the name of that file, for reference.

It will be seen that in every case the codebook that best compressed an author's file was the one developed from samples of the same author's work. Thus using the best-matching codebook as a pointer to authorship does work in this case[6].

[6] This procedure does not work columnwise (unless residuals from the row means are used instead of raw scores). Here it happens that the differences between rows (i.e. compressibility of text files) are significant (1-way ANOVA: $F_{3,12} = 8.47$, p=0.003); while differences between columns (i.e. compression ability of codebooks) are not (1-way ANOVA: $F_{3,12} = 0.33$,

There also appears to be an inverse relationship between the overall ease with which an author's work can be compressed and the overall effectiveness of the codebook produced from that author's work. The author whose work is, on average, hardest to compress of these four is John Pudney: his codebook gives, on average, the best compression of the four. Conversely the author whose work is most easily compressed, on average, is Dylan Thomas: his codebook gives, on average, the least compression of the four. Most probably, Dylan Thomas's work, at the lexical level, contains the most redundancy of these four authors. This makes his file most easy to compress, but reduces the versatility of his codebook for compressing other files. (Following up this sort of speculation would divert this chapter from its main objective, but it does show that cross-codebook compression may have more than a single benefit to offer as a text-analytic technique.)

It is worth noting here that the division into early and late work, though no such partition can be definitive, was not made capriciously. Both Bob Dylan and John Pudney survived serious motorcycle accidents (in July 1966 and February 1967 respectively) to produce subsequent work in a noticeably different style from what they had written beforehand, and in their cases the division is into poems written before and after their accidents. In the case of Dylan Thomas the dividing date is 1937, the year of his marriage -- another natural watershed as he published no poems between 1936 and 1939, when his third book came out. According to Spender & Hall (1970): "these poems announce a new attitude to writing". In T.S. Eliot's case the file TE.1 contained work up to and including *The Waste Land,* published in 1922; the TE.2 file contained work written after that date. Again, the change of style in Eliot's poetical output after the mid-1920s has been widely commented on. In the words of Pearce (1967):

> "Apart from [fragments], he wrote no poetry for publication between 1922 and 1927. In that year, he became a British citizen and was confirmed in the Church of England. The poetry which follows these steps is of a different tone from anything that precedes it".

The point of this brief digression into literary history is to emphasize that the program's task was not made artificially easy: for each of these four writers the difference in style between early and late work was significant. Even so the simple attribution rule "assign each text to the author whose codebook gives the most compression" would have succeeded four times out of four. On a Null Hypothesis that there is no relationship between code-book compressibility and authorship the probability of such a result would be 1/4 to the power of 4, or 1/256.

p=0.802). It is not clear why this should be so, but it means that if differences between rows are not, in effect, discounted, by comparing within rows, then the procedure will not work properly.

However, it is one thing to reject the Null Hypothesis that digram codebooks are completely useless in this respect; it is quite another to establish just how useful they might be in practice. As a step in that direction a second test was performed, this time on smaller text files (an overall aim of this project being to find, if possible, methods able to classify relatively modest amounts of text).

3.3.2 A Second Test on English Poetry

Once again four poets were compared, the same four except that T.S. Eliot was replaced by myself. This was not just egotism, because John Pudney was, for a time at least, my poetic mentor, and I felt therefore that the distinction between Richard Forsyth, John Pudney and Dylan Thomas (whose work Pudney admired) would be a relatively hard one for a program to make. (Other considerations were ease of access and lack of any potential copyright problems.)

For this test, 13 text files were extracted from the works of these four authors and then an 80-item code book formed from what remained. Each of the four codebooks was then used to compress all 13 files.

The results are presented in Table 3.3.

Table 3.3 -- Four Poetic Codebooks Applied to Short Texts.

Code Book → Text File ↓	BD (97963 bytes, 19008 words)	DT (58983 bytes, 10991 words)	JP (41700 bytes, 7408 words)	RF (38453 bytes, 6933 words)
BD.65 (3662)	2265 0.6185	2320 0.6335	2319 0.6333	2272 0.6204
BD.66 (2937)	1742 0.5931	1812 0.6170	1769 0.6023	1745 0.5941
BD.70 (3810)	2282 0.5990	2365 0.6207	2388 0.6268	2305 0.6050
DT.32 (1616)	979 0.6058	944 0.5842	975 0.6033	969 0.5996
DT.35 (6413)	4002 0.6240	3837 0.5985	3965 0.6183	3968 0.6187
JP.67 (990)	612 0.6182	612 0.6182	611 0.6172	617 0.6232
JP.69a (1302)	785 0.6029	790 0.6068	772 0.5929	780 0.5991
JP.69b (6062)	3913 0.6455	3923 0.6471	3892 0.6420	3827 0.6313
JP.69c (5096)	3137 0.6234	3208 0.6295	3118 0.6119	3175 0.6230
RF.66a (506)	311 0.6146	303 0.5988	308 0.6087	303 0.5988
RF.66b (904)	560 0.6195	566 0.6261	553 0.6117	552 0.6106
RF.69a (1006)	624 0.6203	625 0.6213	620 0.6163	605 0.6014
RF.69b (962)	587 0.6102	593 0.6164	572 0.5946	580 0.6029

In this table the initials indicate the author while the following pair of digits gives the year of composition (for BD, DT and RF) or of publication (for JP). All files contained only a single poem except BD.70 (five songs from *New Morning*, B-side) and JP.67 (two poems from the volume *Spill Out*). In brackets underneath each file name is the uncompressed size of that file in bytes. As before, the compressed file size and the ratio of compressed size to uncompressed size are given in the last four columns.

This time the simple rule "ascribe every file to the author whose codebook gives the most compression" makes two clear mistakes, and there is also a tie, on file RF.66a. These results begin to show the limitations of this approach. As expected, the two outright mistakes are confusions between poems by myself and by John Pudney, one in each direction. Interestingly enough, poem RF.69a (entitled *Variations on a Theme of John Pudney*) which was deliberately inserted in the expectation that it might prove hard to classify, was correctly attributed to me.

The tie on the smallest file, containing a mere 506 characters (92 words), suggests that the technique cannot reliably be pushed down to such small text sizes. This opinion is reinforced by that fact that when the four poems of mine tested (the last four rows in Table 3.3) were combined into a single file that file was compressed from 3378 bytes to 2085, 2090, 2056 and 2043 bytes by the four codebooks respectively -- corresponding to a correct attribution.

The mistake on the second-largest file, JP.69b, cannot be dismissed so lightly, however. It must cast some doubt on the robustness of the method; although 10 `correct answers' out of 12 (ignoring ties) on files ranging down in size to 92 words -- smaller than any previous stylometer would even have considered worth testing -- is by no means disgraceful.

Above all, the question arises: just what does a difference in compressed size of only **1 byte** out of 990 (on file JP.67) really signify? This question must be addressed in evaluating cross-codebook compression as a text-classification technique, but its consideration will be deferred till section 3.4 in order to examine some further evidence bearing on the performance of this method, in a slightly wider range of domains.

3.3.3 A Content-Based Discrimination

A third test was carried out in which cross-codebook compression was tried on a problem other than authorship. Here the data was technical English prose, collected as a by-product of the literature search for this thesis, from two academic journals -- *Literary and Linguistic Computing*, published by Oxford University Press, and *Machine Learning*, published by Kluwer. The object was to distinguish between two different styles of modern technical writing. As over 125 different authors contributed to the two samples, it is certainly not an authorship problem. It might be said to be a matter of genre, but it is probably most accurate to call this a content discrimination exercise. In fact it is intended to stand as a surrogate for the type of news categorization study reported in chapter 2, subsection 2.3.2.

More details concerning these texts appear in the next chapter and in Appendix D. This corpus is a haphazard rather than purely random selection, though it should be stressed that the papers that gained inclusion were not included as a result of any deliberate choice by the author, and may therefore be presumed to be representative as regards content.

Each `article' consists the Abstract plus first paragraph from a paper in one of the two journals, 65 from *Literary and Linguistic Computing* and 55 from *Machine Learning*, published during the period 1990-1994. Individual extracts (comprising Abstract and first paragraph) are termed `articles' here in the interests of brevity.

The data from both journals (labelled LL and ML) was divided into two parts, the first part containing articles from 1990 and 1991 and the second containing articles from 1992-94. In addition, to provide a fuller range of results, the early and late T.S. Eliot text files were included in this analysis (like a `spoiler' in a list of multi-choice answers) giving a simple 6-by-6 design in which 80-item codebooks were developed from these six text blocks and then tested on all six blocks. The results are presented in Table 3.4.

Table 3.4 -- Cross-Codebook Compression on two Technical Journals and T.S. Eliot.

Codebook → Text ↓	LL.1 [1990-91]	LL.2 [1992-94]	ML.1 [1990-91]	ML.2 [1992-94]	TE.1 [early]	TE.2 [late]
LL.1 (74049)	44416 0.5990	45739 0.6177	46029 0.6216	45860 0.6194	47785 0.6454	46632 0.6298
LL.2 (45351)	27450 0.6053	27507 0.6065	28048 0.6185	27883 0.6148	29134 0.6424	28449 0.6273
ML.1 (40351)	24793 0.6096	25148 0.6183	24152 0.5938	24435 0.6008	26331 0.6474	25731 0.6326
ML.2 (68185)	41718 0.6118	42329 0.6208	41319 0.6060	40678 0.5966	44158 0.6476	43167 0.6331
TE.1 (42777)	27994 0.6544	27389 0.6403	28738 0.6718	28214 0.6596	25926 0.6061	26551 0.6207
TE.2 (31907)	19858 0.6224	19588 0.6139	20325 0.6370	20109 0.6302	19060 0.5974	18536 0.5809

In this case every pair of files from the same source (LL, ML and TE) is better compressed by both codebooks from that source than by any other codebook, and in this sense the three categories may be said to be clearly distinguished. Interestingly, in this table one of the row-minima does not lie on the main diagonal. In other words, this is one of the few instances where a text file was better compressed by a codebook created from a file other than itself: file LL.2 is best compressed by the LL.1 codebook.

It may not be surprising that verse by T.S. Eliot can be distinguished with ease from modern technical prose[7], but the distinction between the `house styles' of two learned journals both dealing with aspects of computing is quite a subtle one for a program to make. DIGS may even be said to have found marks of the division between C.P. Snow's "Two Cultures", for it is clear from reading these two journals that, although both deal with computer applications, papers in LL are mostly written by people with an arts or humanities background while those in ML are mostly by authors from an engineering or scientific background.

In view of the success of this experiment, a further test was conducted to find out how well the method would work on shorter text samples from the same two journals. For this trial the year with the largest amount of text from each journal (LL.90 and ML.92) was used to form a codebook of 80 elements and these were applied to each of the other years, as well as a 1987 article from *Machine Learning* that happened to be available. Thus two codebooks were applied to nine test texts, ranging in size from 1288 to over 30000 bytes.

The results are shown in Table 3.5.

[7] except perhaps to a paucity of (principally Parisian) post-modernist professors partial to provocatively paradoxical pronouncements.

Table 3.5 -- Codebooks from two Technical Journals Applied to Samples from the Same Journals.

Code-book → Text file ↓	LL.90 (43397)	ML.92 (53550)
LL.91 (30647)	18825 0.6143	19137 0.6244
LL.92 (25653)	15741 0.6136	15697 0.6119
LL.93 (18410)	11291 0.6133	11398 0.6191
LL.94 (1288)	817 0.6343	831 0.6452
ML.87 (1429)	906 0.6340	870 0.6088
ML.90 (28030)	17226 0.6146	16731 0.5969
ML.91 (11214)	6944 0.6192	6874 0.6130
ML.93 (9791)	6186 0.6318	6024 0.6153
ML.94 (4844)	3043 0.6282	2967 0.6125

Once again the size in bytes of each text file is shown in brackets under that file's name. Note that files LL.94 and ML.87 each contain only a single article.

In this table there are eight cases where the same journal's codebook gave greater compression and one (LL.92) where the other journal's codebook gave greater compression. Roughly speaking, that can be summed up as eight correct answers out of nine. Since no previous researcher has attempted this particular classification, the most we can conclude is that the method gives respectable but not error-free performance on a non-trivial content-based text categorization task.

For a final test, therefore, the method was applied to a text classification task that has been intensively studied by previous researchers, the discrimination between Alexander Hamilton and James Madison, co-authors of *The Federalist Papers*.

3.3.4 The Federalist Papers Revisited

For this trial, an electronic copy of the text of the complete *Federalist Papers* was obtained from Project Gutenberg, at Illinois Benedictine University, comprising just over 1.2 megabytes. Then 80-item codebooks were first formed on text files containing undisputed papers by both main *Federalist* authors:

Hamilton codebook formed from 12 papers:	
Nos. 1, 21-29, 60 & 70	(156 kilobytes);
Madison codebook formed from 9 papers:	
Nos. 40-48	(152 kilobytes).

Then these codebooks were applied to all the remaining five Madison papers, to all the twelve disputed papers, and to a random selection of twelve papers by Hamilton not used in forming his codebook (nos. 7, 11, 15, 16, 17, 30, 32, 33, 35, 36, 69, 80).

The results appear in table 3.6 in summary form.

Table 3.6 -- Compression Results on a Sample of Federalist Papers.

Greatest Compression by → Written by ↓	Hamilton Codebook	Madison Codebook
Hamilton	12	0
Madison	5	0
Disputed (probably Madison)	4	8

Compressed sizes of individal papers have not been listed here, just counts showing the number of papers in each of the three categories that were more compressed by either Hamilton's or Madison's codebook.

Here the discriminating effect appears to have been swamped by the strong tendency of Hamilton's codebook to give better compression than Madison's, an interpretation lent weight by the fact that all five of Jay's papers were better compressed by Hamilton's than Madison's codebook.

An alternative way of looking at these results is shown in Figure 3.1, which plots each of these 29 papers as a point in 2D space. The vertical axis is the compressed size (as a proportion of original size) obtained from the Hamilton codebook; the horizontal axis is the compressed size obtained from the Madison codebook. Point labels (H, M and D) stand for Hamilton, Madison and Disputed papers respectively.

[Figure 3.1 -- Scatter Plot of Hamilton, Madison & Disputed Papers: See last page of chapter.]

From this graph it can be seen that the Hamilton and Madison papers are not well separated: most of them lie close to a line rising at approximately 45 degrees from left to right, indicating that the less compressible a file is by the Madison codebook the less compressible it is also by the Hamilton codebook -- and to roughly the same degree. However, most of the disputed papers depart from this general trend. They appear to be, relatively, worse compressed by Hamilton's codebook than by Madison's, an impression that is confirmed by a one-way analysis of variance showing a significant difference between the three groups ($p < 0.001$) in compression by Hamilton's codebook, attributable not to a difference between Hamilton's papers and the other two groups but to a difference between Hamilton's and Madison's undisputed papers taken together and the disputed papers.

It is in fact possible to draw a straight line on this graph dividing all but one of the Hamilton papers from all but one of the Madison papers. If this is done, every single disputed paper falls on the Madisonian side of the line, which would presumably surprise anyone who still believes that Hamilton wrote the disputed papers. Another fact worth noting is that four of the five Madison papers and all 12 of the disputed papers have a size ratio that is larger than the median for Hamilton's papers (0.5628) when compressed by Hamilton's codebook. Overall, these results can be summmed up by saying that, as far as compressibility is concerned, the disputed papers are more Madisonian even than Madison's undisputed papers. Indeed if the vital calibration step of testing the method on five of Madison's undisputed papers had been omitted, this analysis might have appeared to give quite forceful evidence against Hamilton's authorship of the disputed papers.

While these results are undoubtedly interesting and make a minor addition to the pile of evidence in favour of Madison's authorship of the disputed papers, they are much less conclusive than the earlier results of Mosteller and Wallace. The very need for a *post hoc* analysis merely highlights the failure of cross-codebook compression, in its basic form, to give a clear-cut discrimination in this case.

There is certainly some connection between compressibility using digram-based codebooks and authorship, but this experiment puts paid to the idea that cross-codebook compression could serve as a general-purpose `industrial strength' text categorization method. These results are frankly disappointing -- the more so in that several subsidiary analyses, not reported here, were carried out in attempt to rescue the method (e.g. using Chi-squared and log-likelihoods derived from digram frequencies as indices of matching between codebook and text) none of which proved wholly satisfactory.

One conclusion that can be drawn with confidence from this experiment is that Moseller & Wallace (1984) were fully justified in saying that

> "Hamilton's and Madison's styles are unusually similar: new problems, with two authors as candidates, should be easier than distinguishing between Hamilton and Madison."

3.4 Critique

What has been achieved by this investigation?

On the credit side it can be said that cross-codebook compression using progressive concatenation of digrams is a novel stylometric technique which is relatively easy to explain and implement. In fact this method does fulfil five of the seven desiderata listed in chapter 2, subsection 2.4.3:

(1) It can be applied to more than two categories without special effort;

(2) It can be used on relatively short stretches of text, shorter than considered worth tackling by most previous stylometers;

(3) It is not confined to authorship attribution, but can be used in other text classification tasks;

(4) It finds its own markers without having to be told what to look for;

(5) It gets away from over-reliance on words as the main or sole indicators available for text classification, by finding subwords, words and collocations, and hence could in principle be used on non-linguistic material such as DNA sequences.

Of these, the last two points are probably the most significant, but they are not the only advantages of the method. Some of the more important of these additional advantages are listed below:

(6) It requires no special pre-processing of input texts;

(7) It takes as fundamental the **sequential** nature of text, a fact rather under-exploited by much previous work in text classification;

(8) It draws attention to the importance of building a **model** of the texts being analyzed as part of any classification process.

Lastly it is worth mentioning in passing that, although this was only a secondary objective, compression rates achieved using digram codebooks are competitive with those of serious text-compression techniques. For example, Witten et al. (1987) quote the compressed size of a 100,000-byte text file after processing by `compact', a Unix utility program using adaptive Huffman coding, as 57,781 bytes, and after processing by their own arithmetic coding program as 57,718 bytes. Compressed sizes of under 55% were easily attained by DIGS on English text using 120-item codebooks. More recently, Oliver (1993) presents as a `programming classic' an algorithm that gave 36.3% compression on a sample of English text, i.e. compressed a text file to 63.7% of its original size.

Undoubtedly digram codebooks carry useful information and the investigation of their behaviour does represent a contribution to the field of text analysis; however, as was particularly evident from the test on the *Federalist* texts, the method outlined in this chapter does have serious weaknesses:

(1) It gave poor results on a difficult problem (the *Federalist Papers*);

(2) It tended to perform relatively poorly on small text segments, thus negating one of its potential advantages;

(3) It is not well integrated into a coherent inferential framework, such as Bayesian or classical statistical inference, which means that a difference of 1 or 20 or 50 bytes in compressed file size cannot be translated into something more meaningful like a posterior probability or likelihood ratio;

(4) It employs a `knowledge representation' that is somewhat inscrutable; thus, for example, a codebook does not convey much useful information as a **description** of an author's style.

This last failing has been partly remedied by a program called MARKERS which takes a set of codebooks and produces as output the codebook entries which are most distinctive (either by rank or by frequency) in comparison with the other codebooks in that set. This list of the best N markers (where N is input from the user) is helpful in giving a descriptive view of what differences have been found between text types. An example of this program's output, when applied to codebooks from Bob Dylan, Dylan Thomas, John Pudney and T.S. Eliot respectively, is shown below as Table 3.7.

Table 3.7 -- Substrings Characteristic of Four Poets.

```
C:\PC\TA\MARKERS.EXE started on Wed Jul 27 10:04:46 1994
4 codebooks used from poex.cbs   Freq-mode = 0 (Ranking)

Best 16 markers from 80 in codebook 0 bd.cb
 1  147  40.2500 i_
 2  152  39.5000 a_
 3  159  36.7500 ll_
 4  156  33.7500 you_
 5  164  33.0000 in'
 6  137  32.5000 you
 7  167  30.7500 en_
 8  162  28.5000 wa
 9  172  23.5000 's_
10  155  23.5000 to_
11  144  21.7500 .\
12  141  21.5000 o_
13  163  19.2500 _m
14  160  17.7500 om
15  185  17.2500 ee
16  186  16.5000 st_

Best 16 markers from 80 in codebook 1 dt.cb
 1  157  37.5000 to
 2  130  35.0000 _th
 3  156  31.7500 ro
 4  166  31.5000 _h
 5  161  30.2500 ir
 6  170  28.5000 _b
 7  154  28.2500 _w
 8  171  27.5000 ck
 9  174  25.5000 and
10  158  24.2500 _m
11  176  24.0000 in_the_
12  152  24.0000 \\
13  160  21.5000 lo
14  180  21.0000 _f
15  169  17.7500 sh
16  168  17.7500 ra
```

```
Best 16 markers from 80 in codebook 2 jp.cb
 1  141  35.0000 \\
 2  131  34.0000 _th
 3  168  30.0000 with
 4  166  28.0000 r_
 5  165  26.0000 ed
 6  157  25.5000 wi
 7  156  23.7500 of_
 8  167  22.7500 e\
 9  142  20.7500 _s
10  181  20.2500 or_
11  180  18.2500 ur
12  160  18.0000 ti
13  152  17.5000 you
14  185  17.2500 is
15  158  16.2500 ch
16  159  15.7500 .\\_

Best 16 markers from 80 in codebook 3 te.cb
 1  157  32.2500 ha
 2  162  30.2500 \_
 3  171  26.0000 _d
 4  163  24.2500 re
 5  154  24.0000 ti
 6  159  23.2500 _w
 7  173  20.7500 be
 8  176  19.5000 no
 9  132  19.0000 the_
10  156  18.2500 _a
11  150  15.7500 .\
12  147  15.5000 o_
13  182  14.0000 it
14  165  13.5000 to_
15  161  12.2500 is_
16  192  12.0000 se
```

[Here again underscore ('_') has been used to highlight the space character, and the newline character is represented by backslash ('\').]

From this we learn, for instance that Dylan Thomas is comparatively fond of the phrase `in the ' (which appeared already in Table 3.1) as well as the substring `and', while John Pudney is a relatively heavy user of `with'. It should be noted that these last two are more accurately called strings or substrings than words: the word ` and ' is a major reason for the high frequency of `and' in Dylan Thomas, but it also appears in `sand', `hand', `mandrake' and other contexts. Likewise the frequency of `with' in John Pudney is not solely accounted for by its appearance in the word ` with ': `with' also appears within `within', as well as in `without', `withdrawal', `withered' and so on.

The presence of non-alphabetic characters, such as `\\' and `.\\_', may cause some disquiet, as punctuation and layout are very sensitive to editorial and other outside intervention. For this reason, in subsequent chapters texts will be subject to a slightly higher degree of pre-processing, designed to reduce the effect of layout and, to a lesser degree, of punctuation -- although classification with minimal precoding will still be retained as an ideal.

Returning to weakness number (3) above, the lack of a method of assessing the evidential force of differences in compressed size, this could perhaps be cured by using extensive cross-validation (Efron & Gong, 1983; Weiss & Kulikowski, 1991) in which repeated subsamples were used for building a codebook and then testing its compression performance. A program that did this could give estimated standard deviations for compression ratios alongside each codebook, which could be used in statistical inference in the normal way. The cost, however, would be a loss of the simplicity and speed of the method -- two of its chief virtues -- consequently that path will not be followed in the present work.

The most crucial weaknesses, (1) and (2), concern accuracy. Clearly a good classification method should be accurate, and, in the light of the test on the *Federalist* data, the form of cross-codebook compression used here must therefore be abandoned as a general-purpose text classification method. It might, however, have a useful role to play in finding marker patterns that could then be passed as indicator variables to be used by a conventional classifier such as a discrimination tree, a linear discriminant function or a neural network; and this aspect of the method will be followed up in subsequent chapters.

Mention of accuracy raises an important question "what constitutes good performance?", the answer to which is not as obvious as might first be thought, and which motivates the search for a more extensive suite of benchmark tests than used so far. This issue will be further dealt with in the next chapter.

In conclusion, text classification by cross-codebook compression exhibits many desirable features but is marred by some serious weaknesses, outlined above. The rest of this thesis is an attempt to remove or reduce the effect of those weaknesses while preserving as many of the strong points as possible.

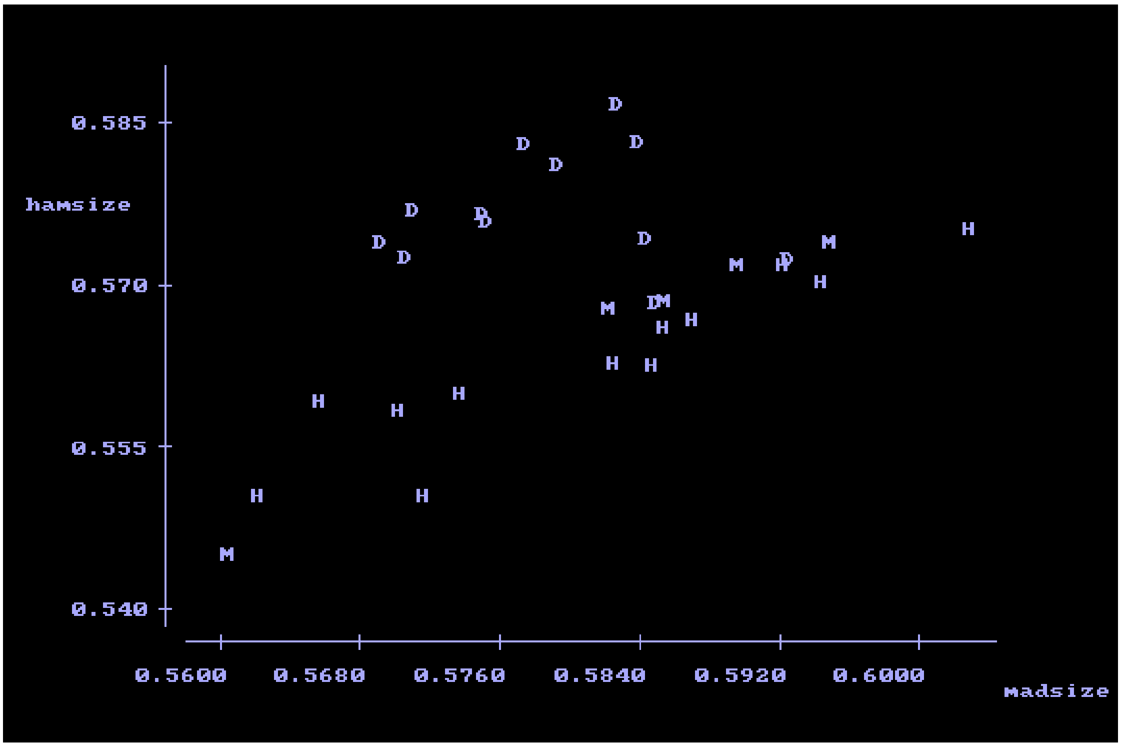


Figure 3.1 -- Scatter Plot of Hamilton, Madison & Disputed Papers.

Chapter 4

TEXT CLASSIFICATION WITH DIAGNOSTIC DIGRAMS

> "Sadly, we have to say that the word is not a clearly definable linguistic unit. We shall, perhaps, have to recognize some kind of unit that corresponds closely to the written word and define it ultimately in terms of a combination of the features we have been considering, though as we shall see in the next chapter some theorists have decided to do without the word altogether. But what the word is or is not depends ultimately on our **total** view of grammar". -- Frank Palmer (1971).

The evaluation of cross-codebook compression in the previous chapter was relatively informal: the method was implemented, applied to a few test cases and the results noted. This may almost be said to be standard practice in computational stylometry. It is rare in the literature for a newly proposed method to be tested on more than two different sets of texts. Indeed application to just one discrimination problem within a single domain is quite normal. For instance, Mosteller & Wallace (1984) concentrated almost entirely on two authors of 18th-century English prose, namely Hamilton and Madison; while, more recently, Matthews and Merriam (Matthews & Merriam, 1993; Merriam & Matthews, 1994) considered only the problem of discriminating plays written by Shakespeare from those of two other Elizabethan dramatists, Fletcher and Marlowe.

While understandable, this practice has a number of unsatisfactory aspects. Firstly, the selection of test cases is generally very narrow. Secondly, statistics other than success rate (or, equivalently, misclassification rate) are rarely reported, although other measures are arguably more sensitive. Thirdly, segmentation of texts into sub-units is typically done according to different criteria for each fresh problem. For example, Mosteller & Wallace (1984) took whole essays as their unit of analysis; Thisted & Efron (1987) analyzed some single poems alongside some groups of poems (grouped at their discretion) of various lengths; while Elliott & Valenza (1991) reported results based on breaking a poet's corpus into 500-word chunks. Other researchers have had other ideas about what constitutes a natural subdivision of text into shorter units. For these reasons comparison between different text-classification schemes is fraught with difficulties.

In this chapter an attempt is made to address these issues by establishing a common framework within which different methods of automatic text categorization may meaningfully be compared. As a starting-point, this framework is then used to evaluate the performance of both a string-based and a word-based Bayesian text classifier.

4.1 Towards an Empirical Framework

A major aim of this thesis is to compare different algorithms, and/or representation schemes, for text classification. Therefore it was felt necessary, before proceeding further, to establish a `level playing-field' on which they could compete. Furthermore, since the kind of data stored as text is so multifarious, it was concluded that no purely theoretical analysis would suffice. These two considerations suggested the need for some sort of benchmark suite of representative test problems on which the various methods developed during this project could be tried.

The use of benchmark cases -- for testing the compliance of compilers with programming-language definitions, for comparing the `throughput' of systems from different suppliers, and so on -- is an accepted part of computer science. Benchmarking has many benefits, but also, of course, some drawbacks. The main problem with benchmark suites, particularly those meant to test processing speed, is that, once they become well established, it is possible for manufacturers to over-optimize their equipment with respect to performance on widely published benchmark problems at the expense of other capabilities[8]. However, this possibility does not pose an immediate risk in the present context as the benchmark suite compiled here is, of necessity, novel.

Its novelty is enforced by the fact that no benchmark suite suitable for the present purpose currently exists, as far as I have been able to ascertain. Large collections of public-domain data-sets do exist, such as the Oxford Text Archive at Oxford University, the Project Gutenberg Text Database at Illinois Benedictine College, and the Repository of Machine Learning Databases at the University of California at Irvine. These constitute valuable resources, but do not meet the present need, for two different reasons. The Oxford Text Archive holds over a gigabyte of text in electronic form, containing more than 1300 titles. It is a purely passive repository, however, and its contents have not been assembled according to any guiding principle other than availability. The archive's administrators do not require submitted texts to conform to any given standard of correctness or formatting: some have been extensively tagged with analytic codes, while others are essentially plain texts. In addition, the texts vary greatly in their accuracy. Similar remarks apply to the files held by Project Gutenberg. Thus neither of these archives constitutes a benchmark suite in the normal sense of that term, although they hold plenty of materials which could form part of such a suite.

[8] For example, the time taken to find all prime numbers less than 20,000 using the `Sieve of Eratosthenes' algorithm used to be quite a good index of the efficiency of machine-code produced by different C compilers; but once the test became widely quoted in software reviews, compiler vendors started to optimize their code-generation procedures for this particular problem, rendering it useless as a test case.

The machine-learning repository at UCI, on the other hand, has been collected with the explicit purpose of facilitating comparative testing (Murphy & Aha, 1991). The problem with this source is that its data-sets are almost entirely numeric -- held predominantly as flat files in which rows encode cases as numeric feature-vectors. An additional problem with this database has been pointed out by Holte (1993) who found that its problems do not span a wide range of difficulty. Most are, in fact, rather too easy, in that good classification performance can be achieved by simple rules, such as 1-level decision trees that test only a single attribute.

What is required for the present purpose is a hybrid which is collected with benchmarking in mind, like the UCI repository, but which is textual in form, like the Oxford Text Archive or Project Gutenberg. Accordingly, part of the effort involved in this project was expended in assembling an electronic anthology from which a set of test problems could be extracted. These test cases will be designated collectively from here on as TBS1 -- Text Benchmark Suite (Version 1). It is hoped that this collection (or later versions of it) will in due course be made publicly available and thus that a spin-off from the present work will be a resource that fills the gap between existing text archives and machine-learning databases.

4.2 Selection of Test Problems

In choosing texts for inclusion in TBS1, several considerations were borne in mind. The prime requirement of such a benchmark suite is that the true category of each text be known. Thus if the problem concerns authorship attribution, the true authors of the texts should be uncontested; if the problem concerns chronology, the actual date of composition should be known with reasonable certainty.

This criterion may seem obvious, but it excludes several celebrated cases (such as Plato's dialogues and Shakespeare's plays) where the dating and/or authorship is controversial.

4.2.1 Selection Criteria

Thus the problems in TBS1 were selected according to a number of requirements:

1.Certainty -- the provenance of each text should be well-attested;
2.Variety -- problems other than authorship should be included;
3.Language -- not all the texts should be in English;
4.Difficulty -- both hard and easy problems should be included;
5.Size -- the training texts should be of modest size such as might be expected in practical text classification tasks.

The last point may need amplification: although many huge text samples are available, most text-classification problems in real life require classification to be made on the basis of training samples in the order of thousands or tens of thousands, rather than hundreds of thousands or millions, of words. An enormous sample of well-attested training text is, therefore, something of a luxury. It was thought important that the classification methods tested here should be able to perform reasonably well without reliance on this luxury.

Subject to these constraints, a Baker's Dozen of test cases was compiled. Although it cannot claim to be a random sample (since the population of interest is ill-defined) and although it is inevitably a tiny fraction of those that might have been chosen, it is intended to constitute a representative selection of some of the more important types of text-classification problem encountered in practice. It contains five authorship problems, four chronology problems, one problem of content or subject-matter, and three miscellaneous text-categorization tasks. Each is briefly described below under the relevant heading, labelled with the short mnemonic used to refer to it in later sections. (For fuller information on sources, editing conventions and so forth, see Appendix D.)

4.2.2 The Text Benchmark Suite

<u>Authorship / English Poetry</u>

LIT1 (4 classes) -- Poems by Bob Dylan (about 21,000 words), Dylan Thomas (about 12,000 words), John Pudney (about 8000 words) and Richard Forsyth (about 6000 words).

LIT2 (3 classes) -- Poems by Helen Forsyth (about 5500 words), James Forsyth (about 6000 words) and Richard Forsyth (about 6000 words), i.e. my mother, my father and myself[9].

LIT3 (3 classes) -- Poems by Ezra Pound (about 8000 words), T.S. Eliot (about 10,000 words) and William B. Yeats (about 12,000 words). (*The Waste Land*, which was written by T.S. Eliot but heavily amended on the advice of Ezra Pound (see Valerie Eliot (1971)) was **excluded** from this test.)

As is usual in machine learning the data was divided into <u>training</u> and <u>testing</u> sets. This division was made by allocating **files** randomly to test or training sets until at least 30% of the available corpus of each author had been assigned to the test set, any remainder going in the

[9] The Forsythian bias will perhaps be considered an excusable indulgence, as a reward for the time and effort required to type, collate and/or edit several megabytes of text. In any case, it seemed a good idea to include a case of discrimination between related authors, and I wasn't able to obtain enough of the Brontë sisters in time. (They will have to wait till version 2.)

training set. As this data is held (with a few exceptions) in files each of which contains writings composed by one author in a single year, this mode of division meant that works composed at about the same time were usually kept together; and, even more important, that single poems were never split between test and training files. The effect of this file-based blocking is presumably to make these tests somewhat more stringent than random allocation of individual poems to test and training sets would have been.

(It should be noted that for the repeated author, myself, the test-training division was different in LIT1 and LIT2.)

Authorship / English Prose

FEDS (3 classes) -- The same selection of **undisputed** papers by the three *Federalist* authors Hamilton (approximately 53,000 words), Jay (about 8000 words) and Madison (about 39,000 words) as studied in Chapter 3. (Original texts from Project Gutenberg.)

Here the division into test and training sets was identical to that described in subsection 3.3.4.

Authorship / Snobol4 Programming Language

SNOB (4 classes) -- Samples of program source code written by four different programmers in the Snobol4 programming language. These were Richard Forsyth (about 39,000 bytes), James Gimpel (about 100,000 bytes), Ralph Griswold (about 62,000 bytes) and Mike Shafto (about 116,000 bytes). My own programs were mostly written in 1994, for this project, with one program from 1979 and another from 1988. The other three sets came from diskettes purchased from Catspaw Inc. in Salida, Colorado, USA, who sell them as useful program libraries. (This problem is the only one to fulfill the requirement that not all texts should be in the English language; indeed it is not even in natural language, although since the programs were written by English-speakers, a fair proportion of natural English is included, as comment lines.)

Division into test and training sets was made by concatenating each author's files, in the order that they appeared on the distribution disc, and then dividing roughly at the half-way point, always at a program or function ending.

Chronology / English Poetry

AGE1 (2 classes) -- Songs by Bob Dylan, written before and after his motorcycle crash of July 1966. (See chapter 3.)

AGE2 (2 classes) -- Poems by Emily Dickinson (about 8000 words), early work being written up to 1863 and later work being written after 1863. Emily Dickinson had a great surge of poetic composition in 1862 and a lesser peak in 1864, after which her output tailed off gradually.

AGE3 (2 classes) -- Poems by James Forsyth, written up to 1945 (early work) or from 1985 onwards (later work). Even without the family connection it would be interesting to have such a clear-cut example of a division into two productive periods, separated by an unusually long, 40-year, span during which the author wrote very little poetry.

AGE4 (2 classes) -- Early and late poems of W.B. Yeats. Early work taken as written up to 1914, the start of the First World War, and later work being written in or after 1916, the date of the Irish Easter Rising, which had a profound effect on Yeats's beliefs about what poetry should aim to achieve.

For these four problems the classification objective was to discriminate between early and late works by the same poet. The division into test and training sets was once again file-based: in these cases the files of each poet were ordered chronologically and assigned alternately (i.e. from odd then even positions) to two sets. The larger of the two resulting files was designated as training and the smaller as test file.

Content / Technical Prose

MAGS (2 classes) -- This used articles from the two journals *Literary and Linguistic Computing* (about 20,000 words) and *Machine Learning* (about 19,000 words). The task was to classify texts according to which journal they came from. (See subsection 3.3.3.)

Here the training set for *Literary and Linguistic Computing* consisted of the years 1990 and 1991, with the test set being the articles from 1992, 1993 and 1994. For *Machine Learning* the training set comprised the two largest files, namely those of 1990 and 1992; the test set was from years 1987, 1988, 1989, 1991, 1993 and 1994. (The files for years 1987-89 each contained only a single article.)

Persona / English Prose

WAVE (3 classes) -- This problem was taken from the novel *The Waves* by Virginia Woolf (1931, 1992), the text of which was obtained from Royal Holloway College in a form that enabled convenient extraction of passages by individual speakers, as well as the narrator. (The original source was the Oxford Text Archive, but without speaker-identification tags[10].) This book consists of

[10] All such tags, including and especially speaker-identifying tags, were of course removed

monologues or soliloquies by six different characters, linked by descriptive passages. The classification task was to distinguish speech by two of the characters in the novel, called `Jinny' and `Rhoda', from narrative text by the author herself. This might be considered a problem of distinguishing between the genres of narrative and dialogue, but, while it is in part a genre problem, differentiating betweem two fictional character's speech is something else. In any case the book is so peculiar that this problem is probably better characterized under some other label, as done here.

Test/training sets for WAVE were made (for each of the three categories) by dividing at the first paragraph break after 400 lines by the speaker or narrator concerned and putting the text up to that point into the training file and the rest into the test file.

Syntax / Artificial Data

ART1 (2 classes) -- Output from two different `poetry generation' programs, one re-written in C by myself from an earlier version in Basic (Forsyth, 1978) and another written in Snobol4 by James Gimpel (Gimpel, 1986).

ART2 (2 classes) -- Output from two different versions of my random `poetry' generator: version 1 with the grammar as in the original (Forsyth, 1978) and version 2 with identical grammar except for the addition of relative clauses in the form `that' + Verb-Phrase allowed at the end of Noun Phrases with a 15% probability. (The use of `that', which was already in the lexicon as a determiner, to introduce a relative clause rather than `which' avoided animate/inanimate discord and also meant that the vocabularies of the two systems were **exactly the same**, unlike in ART1 -- although relative frequencies among words differed.)

Here test and training sets were obtained by running the programs to produce the required amount of text (about 6400 words) each time re-seeding the pseudo-random number generator.

To give an idea of what sort of `language' is being discussed here, specimens from the Forsyth and Gimpel generators are given below in Table 4.1, a complete compostion from the former and two verses from the latter. They were chosen randomly and are completely unretouched.

from the texts before inclusion in TBS1. In addition, the forms "said Jinny" and "said Rhoda", which were invariably used in the first sentence of a passage by each of the speakers concerned, were altered to "said x".

Table 4.1 -- Specimen Output from Two Computer Poets.

Forsyth Generator:

```
        looking at Venus

  even the gentle and desolate dusk
    frowns.
  that world
    is of every love's still light on some winter.
  despair
    would yearn.
  speech without nature
    may grow.
  the feeling of my fiercely bright look
    could deceive her child.
  time
    may be his gold.
  your coin
    unmakes your vision.
  some despair
    thinks.

  every chosen thief
    makes paradise behind the diabolical and endlessly deep sky.
  truth
    can control his evening.
```

Gimpel Generator:

```
A pensive trap should simmer on the hunger of land
And should beseech the action of energy.
But a petal of philosophy that gurgles pursues power's taste
While the hunger of land pursues that pensive trap.

A gentle cover must think beside a timorous spark
And must become a frail muffin.
But charity beside night ensnares earth of thought
While a timorous spark strengthens that gentle cover.
```

The great advantage of artificial data (apart from its ease of generation) is that the sources are well-specified and can be manipulated to have particular characteristics. Thus the reason for success or failure of a classifier in distinguishing between data from different sources is normally readily apparent. The great disadvantage is that a classifier's performance in precisely controlled conditions does not necessarily foretell what it will do with the sort of messy data found in the real world. Both types of data have their place in a balanced benchmark suite.

4.2.3 Replication Policy

It should be noted that the same test/training division will be used for each of these 13 problems in all tests subsequently reported. For the purposes of accurately estimating a classifier's likely future error rate on a particular problem it might well be better to use repeated random subsampling, as in bootstrapping or cross-validation (Efron & Gong, 1983); but for comparing several classifiers over a range of data sets it is more convenient to have a fixed, standardized set of training and test cases.

4.3 Preprocessing and Other Preliminaries

As already stated, all files in TBS1 are plain ASCII text with no embedded codes, no diacritical marks and no accented letters.

4.3.1 Formatting

In order to impose some uniformity of layout and thus reduce the effect of factors such as line-length (usually not an authorial decision and in any case very easy to mimic) all texts were passed through a program called PREP before being analyzed. This program made some minor formatting changes: tabs were converted to blanks; runs of multiple blanks were converted to single blanks; and upper-case letters were converted to lower case. By far the most important change made by PREP, however, was to break running text into segments which were then treated as basic units in subsequent analyses.

4.3.2 Choice of Analytic Unit

A reasonable choice of what constitutes a natural unit of text is by no means straightforward. As stated at the beginning of this chapter, different researchers have made different decisions about the best way of segmenting long texts and thus turning a single sequence of characters into several cases or observations. Some have used fixed-length blocks, e.g. of 500 or 1000 words, others have respected natural subdivisions in the text, e.g. into verses, chapters or poems. Both approaches have merits and drawbacks.

Because linguistic texts have a hierarchical structure, they can be subdivided in several ways: there is no single universally correct segmentation scheme. Some candidate units of analysis are listed below, ranging from the smallest to the largest, along with comments on their suitability as elementary observations for classification purposes.

1.Character
-- well-defined but completely lacking in internal structure.
2.Word
-- surprisingly hard to define, and relatively lacking in internal structure.
3.Line
-- often imposed on a text by someone other than its originator.
4.Sentence
-- very hard to define: poets especially do not write in grammatical sentences.
5.Paragraph/Stanza
-- extremely variable in length.
6.Fixed-length Block
-- arbitrary and artificial, but at least tractable.
7.Chapter/Poem
-- variable in length and typically too large to work with conveniently.
8.Book/Opus
-- plenty of internal structure but much too large to allow sufficient samples for reliable statistics.
9.Corpus/Oeuvre
-- unique and therefore incomparable.

In general the smaller units are too short to provide opportunities for stylistic habits to operate by affecting the arrangement of their internal constituents, while the larger units are too long to provide enough samples for reliable statistical reasoning. Hence the ideal sub-unit of text for classification purposes lies somewhere between the sentence (item 4) and the chapter or poem (item 7).

No choice can be perfect, but as an aim of this project was to advance towards the possibility of classifying shorter text segments than previously considered feasible, it was decided to have a standard unit size towards the lower end of this spectrum and divide all texts into `sonnet-sized chunks' (SSCs). A survey of sveral poets in the electronic anthology (including some not in the TBS1 list, such as Shakespeare) suggested that the mean size of an English sonnet is about 639 characters, excluding line feeds, or roughly 113 words. And so PREP was set to divide the texts into slabs of approximately that length before further analysis.

As a compromise between uniformity and naturalness, PREP is not completely rigid in this subdivision. It has been written to request a target line length, in bytes, from the user and then to look for suitable break-points in the input between 96% and 108% of that line length, in the following order of priority:

1. treble new-lines (typically end of section);
2. double new-lines (typically end of stanza or paragraph);
3. punctuation mark followed by new line;
4. any white space.

Only if none of these four ordered searches suceeded will it, as a last resort, break the line exactly at the target character position. (In fact, this never happened.)

Experiments with *Altarwise by Owl-Light* by Dylan Thomas, a poem consisting of ten linked 14-line sonnets, indicated that a target length of 615 resulted in division into 10 chunks, each containing a whole sonnet[11], and so 615 was used as target length for all texts in TBS1, not just the poetic ones, prior to analysis by the classifiers.

Thus, after preparation by PREP, each file consists of lines of somewhat over 600 bytes and each line, or SSC, is from then on treated as a unit.

4.4 Establishing a Baseline

With the benchmark suite in place, testing could begin. An initial trial was conducted mainly to establish a baseline, i.e. to assess what level of classification accuracy could be expected on this particular collection of problems and to find out which of them were hard and which easy.

For this purpose a simple program called BBTC (Basic Bayesian Text Classifier) was written in the Spitbol dialect (Emmer et al., 1989) of Snobol4 (Griswold et al., 1971). As Wasserman (1993) says: "the Bayesian classifier is often used as a standard against which all other methods are evaluated." BBTC was deliberately kept relatively unsophisticated on the basis that it is wise in any comparative experiment to have a control condition or something equivalent. As the cross-codebook compression study of Chapter 3 had shown quite short substrings (particularly digrams) to hold useful information about text type, it based its decisions on n-gram frequencies, where n could vary from 1 to 4.

4.4.1 A Basic Bayesian Text Classifier

BBTC works as follows. (A full program listing is given in Appendix B.)

[11] Because of the asymmetry in PREP's search, the actual average line length was 639.7 bytes, not 615.

Phase 1

1. Read training file of each category, tallying n-gram frequencies in a hash table.

Phase 2

2. Read test files line by line (1 SSC at a time):

2a. For each line calculate posterior probability of its coming from each source by treating each n-gram as an independent item of evidence in a Bayesian inference process.

2b. Pick the category with the highest posterior probability.

Except for 1-grams (individual characters) the independence assumption is violated, due to overlap; but Gottman & Roy (1990), in their textbook on sequential analysis, report that it is unusual for this violation to make a difference in practice.

The Bayesian inference scheme employed could be described as conventional (Iverson, 1984; Naylor, 1989; Schum, 1994), were it not still regarded by many conventional statisticians as akin to witchcraft. In a nutshell, it is a way of arriving at P(H | E), the probability of a hypothesis H given some evidence E, from P(E | H), the probability of that evidence given the hypothesis. It can perhaps be best explained with the help of an example. Table 4.2 is to be taken as from an authorship problem with three candidate authors, A1, A2 and A3. Workings are shown for two successive digrams (gramsize=2).

Table 4.2 -- Tableau Illustrating Bayesian Reasoning.

Author :	A1	A2	A3
digram = `ae'	211 / 16960	219 / 14485	106 / 26864
p(E | H) =	0.012441	0.015119	0.00367
ln(p) =	-4.3868	-4.1919	-5.6069
digram = `er'	496 / 16960	303 / 14485	232 / 26864
p(E | H) =	0.02925	0.02092	0.00864
ln(p) =	-3.532	-3.8671	-4.7518
log.sum =	-7.9188	-8.059	-10.3587
exp(log.sum) =	0.0003638	0.0003162	0.00003172
P(H | E) =	0.5111	0.4442	0.0446

This table may be viewed a series of snapshots of the program's reasoning in which only two items of evidence are considered. The first item, the digram `ae' occurs 211 times out of 16960 digrams in the training data for author A1. Thus the estimated probability of its occurring if A1 is author of the present line is p(`ae' | A1) = 211/16960 = 0.012441. It occurs 219 times out of 14485 digrams in A2's training text (p = 0.015119) and 106 times out of 26864 in author A3's (p = 0.00367). The conditional probabilities derived from these ratios are converted to logarithms (in the row labelled `ln(p)') because adding logarithms corresponds to multiplying probabilities but is less prone to rounding errors due to floating-point imprecision. The same process is carried out when the digram `er' arrives, which is a somewhat more likely event for author A1 than A2 (0.02925 versus 0.02092) and considerably more likely for A2 than A3 (0.02092 versus 0.00864). The figures in the row labelled `log.sum' are simply the totals of the logarithms of the conditional probabilities. In the BBTC program these sums would have been obtained from some hundreds of digrams, rather than just two, but the logic is the same. By taking the exponent of these sums, the conditional probabilities for each author of this particular sequence of two events are obtained, shown in the penultimate row. These are normalized in the last row of the table to give a posterior probability for each author (P). This is equivalent to applying Bayes's Rule with equal prior probabilities.

In this example, A1 is the most likely author, but not by much, with A3 rather unlikely: both items of evidence were comparatively unfavourable to A3.

Essentially this process implements the fundamental Bayesian formula

$$posteriorprobability \propto priorprobability \times likelihood$$

where likelihood is P(E|H), the probability of some item of evidence given a particular hypothesis, and posterior probability is P(H|E), the probability of that particular hypothesis given the item of evidence (Schmitt, 1969).

BBTC uses exactly this method of computing posterior probabilities, and thereby assigning categories, but with two modifications:

1. An n-gram with zero frequency in the training table is assigned a likelihood of (m / 1.618034) / N, where m is the frequency of the least common n-gram in the training data (usually 1) and N is the total number of n-grams in the training data;
2. the sum of the log-probabilities is actually divided by sqrt(g), where g is the number of n-grams in the current line, before being subject to exponentiation.

The first of these two `fixes' is justified not merely to avoid trying to take the logarithm of zero (which would lead to an arithmetic error) but on the basis that zero probabilities just do not occur. It amounts to saying that the probability of encountering a previously unseen n-gram of any author/text-source is smaller by the `golden ratio' than that of the least likely n-gram previously seen. The second of these modifications is admittedly *ad hoc*. An argument might be constructed on the basis that the standard error of an estimate decreases with the square root of the sample size, but it would be rather shaky. In practice the effect of this modification is to attenuate the more extreme final probability estimates and thus avoid unrealistic results such as 0 or 1. It has no effect on the categorization, only on the certainty attached to that categorization.

4.4.2 Character-Based Results

BBTC was run on all 13 benchmark problems and the percentage of lines (SSCs) correctly classified was recorded for three different sizes of n-grams -- single characters, digrams and trigrams. Results for each test set (i.e. on unseen data) are shown in Table 4.3.

Table 4.3 -- Percentage Success Rates of Bayesian Classifier for Various n-gram Sizes.

	gramsize = 1	gramsize = 2	gramsize = 3	Default
LIT1	68.18	70.91	76.36	39.09
LIT2	83.33	87.04	87.04	37.29
LIT3	71.74	80.43	81.52	27.17
FEDS	47.89	57.57	57.32	62.78
SNOB	65.34	67.33	64.94	36.65
AGE1	61.73	69.14	62.96	60.49
AGE2	83.33	83.33	80.00	66.67
AGE3	76.67	86.67	60.00	66.67
AGE4	80.00	68.00	72.00	56.00
MAGS	69.23	89.74	94.78	40.17
WAVE	57.14	66.67	83.33	47.62
ART1	100.0	100.0	100.0	61.98
ART2	71.29	78.22	75.25	54.46
Mean =	71.99	77.31	76.58	47.45

The mean in the final row is averaged over problems, which corresponds to giving each problem equal weighting rather than each case (SSC). As the number of cases actually available in each test set is not an especially good index of the importance of that problem, this seems a reasonable decision.

The column labelled `Default' gives the percentage success rate that would be obtained by simply classifying each instance in the test set as belonging to the most common category in the training data.

The results of Table 4.3 give us an initial benchmark level: 77% success rate. This can be achieved with quite a simple method; so with more complex techniques we would want to reach 80% or thereabouts on this mixture of problems.

This table also provides some information about the relative difficulty of the various problems within the benchmark suite. It would seem that ART1 is the easiest data to classify and FEDS the hardest. Though ART1 is hardly a challenging problem, in that the two sentence-generating programs can be relied upon for consistently different output, it is handy to have it in the set as a check. A classifier that fails to exceed 99% success rate on this data-set probably has a programming error in it.

MAGS also turns out to be an easy problem. There are superficial characteristics of the two text sources, such as the substrings `ngu' (as in `language' or `linguistic') and `lgo' (as in `algorithm'), which are common in one category and rare in the other. With only a single example of this type, it is unsafe to generalize, but it begins to appear that content-based classification may, in general, be easier than authorship attribution.

The hardest problem, as measured by percentage success rate, is the distinction between the three *Federalist* authors. This confirms Mosteller and Wallace's claim that most authorship problems are easier that the one they chose to tackle. (Incidentally, it should be noted that the inclusion of Jay, and the division into SSCs of just over 100 words, rather than essays averaging around 2200 words, makes this an even more difficult task that the original *Federalist* discrimination problem.)

A two-way analysis of variance on these figures (for main effects only) showed a highly significant main effect of problem ($F_{12,24} = 8.02$, $p < 0.0001$) but no significant effect of n-gram size ($F_{2,24} = 2.22$, $p = 0.13$). However a paired t-test between columns 1 and 2 gave a t value of 2.54 ($p = 0.026$ with 12 degrees of freedom). Comparing these columns it can be seen that in 10 problems there is an increase, in 2 there is no difference and in only one is there a decrease in success rate when moving from 1-grams (single characters) to digrams.

This apparent superiority of digrams tends to support the conclusions of Kjell (1994), who obtained good results in a re-analysis of the Federalist problem by using selected letter-pair frequencies as input to an artificial neural network, and recommended letter-pair frequencies as easily computable stylistic indicators; although it should be borne in mind that he worked on alphabetic data only, with all spaces and other characters removed from the texts, so the present findings are only roughly comparable to Kjell's.

It would be rather extraordinary if there were no improvement from characters to digrams; but perhaps even more extraordinary is the fact that classification based solely on single-character frequencies works as well as it does. This condition was only included for completeness, so success rates such as 80% (AGE4, distinguishing between early and late Yeats) and 83.33% (LIT2, discriminating among three related poets) were quite unexpected.

Ledger (1989) performed a multivariate analysis of work by seven Greek authors, including Plato, Thucydides and Xenophon, using -- among other variables -- frequencies of different letters at the ends of words. It is understandable that final letter-frequencies might be distinctive in an inflected language like ancient Greek; but in an uninflected language like English, using frequencies of single characters, including punctuation marks, without reference to their position in words might well be thought too simplistic to yield useful results.

Ule (1982) briefly mentions experimenting with the 26 letter-frequencies and 10 digit-frequencies in his Marlovian studies, though he does not quote results derived from such experiments, and more recently Ledger and Merriam (1994) have used single letter frequencies as part of an investigation into Shakespeare's versus Fletcher's contribution to *Two Noble Kinsmen*. So it is not completely unprecedented to employ character frequencies as stylometric variables in studies of English texts but, as Ledger and Merrian themselves note, it is most unusual. The relative success here of such a simple-minded approach is in some ways rather disconcerting.

Trigrams do not give consistently better results than digrams. A likely explanation for this is simply that there are many more different trigrams than digrams and therefore the estimated probabilities used by the classifier tend to be based on smaller samples. In addition, zero frequencies are more common, and so the `fudge' of using ((m / 1.681034) / N) as an estimate comes into play more often: this must introduce further inaccuracy.

4.4.3 Results on Some Individual Problems

From a benchmarking standpoint, performances on individual test problems do not matter very much. Nevertheless, to give the reader a clearer view of this benchmarking exercise, results on a couple of individual test cases are singled out here and briefly summarized.

One problem worth a second glance is FEDS, the distinction between the three *Federalist* authors. Here the performance of the classifier appears very poor indeed: it is the only case out of 13 in which the success rate never rose above the `default' value attained by applying a null rule. However the position is not quite so dire as this suggests, firstly because the default figure of 62.78% is misleadingly inflated by the greater number of Hamilton samples available and secondly because the program did after all capture some information about the differences between these three authors. This is illustrated by the dotplots in Figure 4.1, below, which show the distributions of the probability estimates made, using digrams, by BBTC concerning Hamilton's authorship for the three classes of data in this test set.

```
                                     .
                                 :. ::
                           .  :  :: ::.   .
                           :  :  :: :::  ::
Hamilton                : :: .:  :: :::: ::...:  :
                        :.:: ::::::::::::::::::  :.
               .  .     :::::::::::::::::::::::. :::
        :     .:.::.:.: ::::::::::::::::::::::::.:::: : .:. ..
    +---------+---------+---------+---------+---------+-------hamprob

Jay

        .     .. .::.  :   . .::...      .
    +---------+---------+---------+---------+---------+-------hamprob

                                :
                       .   .    :.
Madison               :: . : :..::: :
                :  :.::: : :::::::: :. . ..   .
         . .  .:::::::::.:::::::::: :::: :::.::.. . ..        .
    +---------+---------+---------+---------+---------+-------hamprob
    0.240     0.300     0.360     0.420     0.480     0.540
```

Figure 4.1 -- Distributions of Estimated Probabilities of Hamilton's Authorship in Federalist Test Cases.

Here `hamprob' is the probability assigned by the program to the hypothesis that Hamilton is the true author. Each dot represents a single SSC. It is clear that while the distributions overlap, there is also a clear difference between them, in the desired direction. The confusion matrix for the 403 test cases, given below as Table 4.4, confirms that the classifier is performing at a level better than chance.

Table 4.4 -- Confusion Matrix for Federalist Authors.

Actual :	Hamilton	Jay	Madison
Predicted Ham	166	6	54
Predicted Jay	23	11	20

Predicted Mad	64	4	55

The WAVE problem also has some interesting aspects. The results reveal it to be a composite of an easy problem (telling the narrator from two of her fictional female speakers) and a hard problem (telling those speakers apart). Figure 4.2, below, is a plot of all the test cases on two dimensions. The vertical axis is BBTC's estimated probability, using digrams, that `Jinny' is the correct persona. The horizontal axis is the same estimate for Virginia Woolf herself, i.e. the narrator. The labels J, R and V stand for Jinny, Rhoda and Virginia, respectively.

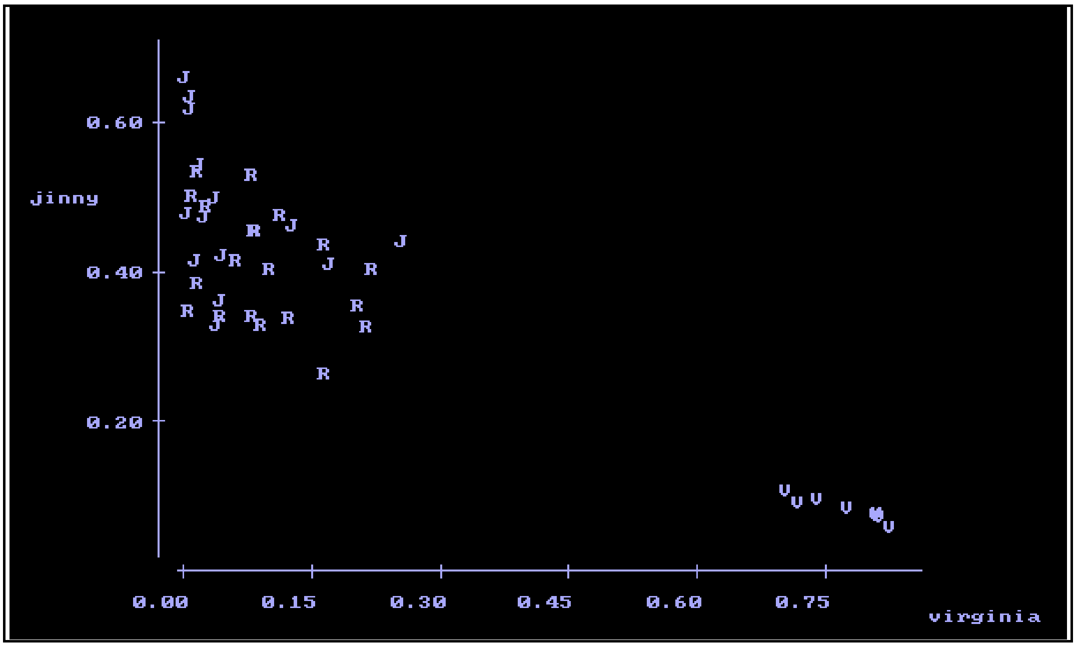


Figure 4.2 -- Plot of Virginia Woolf's Narrative Passages & Passages by two of her Fictional Characters.

This graph reveals a clear separation into the narrative voice of Virginia Woolf on the one hand and her two characters on the other hand. (Since the words `I' and `am' which both characters use quite often are never used in the narrative passages, this is hardly surprising.) The two fictional personae, however, are not very well separated.

4.4.4 Measures Other than Success Rate

Percentage of correctly classified test cases, the only measure of accuracy used so far, is perhaps the most immediately intelligible quantitative measure of a classifier's performance; but it is not necessarily the best in all circumstances. It does not, for example, differentiate a classification based on probabilities of 0.51 and 0.49 from one based on probabilities of 0.99 and 0.01. In other words, it ignores some information that a probabilistic classifier provides.

In fields where evaluation of uncertain forecasts is required, Cooke (1991) recommends using the only proper and relevant scoring rule[12], namely the logarithmic rule:

$$R_i = \ln p_i$$

2where R_i is the `reward' to be given when outcome i occurs to a forecaster who gave p_i as the predicted probability of that outcome. In essence, this is the "logarithmic penalty" scoring function of Mosteller and Wallace (1984).

BBTC was written to compute this logarithmic penalty, using logarithms to the base 2 to facilitate an information-theoretic interpretation, and this will be quoted as an alternative quality measure in what follows.

In addition, since both success-rate and log-penalty are instance-based measures and since such measures are susceptible to arbitrary imbalances in the number of test cases in each category, it was felt useful to have a category-based quality measure as well.

For this purpose a quantity called precall was computed -- a hybrid of the two measures most commonly used in Information Retrieval (Salton & McGill, 1983), namely Precision (the proportion of cases classified in a given category that do belong to that category) and Recall (the proportion of cases in a category that are classified as being in that category). Precall is computed by summing the square root of the product of Precision times Recall for each category and then dividing by the number of categories. It is thus the arithmetic mean, over categories, of the geometric mean of Precision and Recall. It varies between 0 and 1, providing an index of how close the classifier's performance is to the ideal with each category weighted equally.

Both these indices will be quoted in what follows.

4.5 Trials on Word-Coded Data

In view of the findings of Burrows (1992) that rates of usage of high-frequency words (typically the 40 to 80 commonest words in the combined corpus under study) reveal clear patterns of similarity and dissimilarity among authors and genres, it was felt that it would be valuable to perform a word-based replication of the benchmarking exercise reported in the preceding section.

[12] A `proper' reward structure is one in which the decision-maker cannot, in the long run, gain by giving estimated probability values other than those he or she actually believes to be correct; a `relevant' scoring rule is one where the reward given depends only on the outcome that actually happens, not on outcomes that did not, in fact, occur.

BBTC is a character-based program, so this was done by first creating coded versions of all 13 test data sets, in which words were replaced by single characters.

4.5.1 A Form of Word-Coding

To recode the texts, two Spitbol programs called VOCLIST and PRECODE were written. VOCLIST produces a high-frequency word list for a text file or list of text files. PRECODE then uses such a list to encode a sequence of words as a string of characters.

Defining what we mean by a word is problematic (Palmer, 1971), but VOCLIST considers any unbroken sequence of alphanumeric characters as a word -- having first stripped apostrophes from its input. Thus, for example, "don't" becomes the word "dont" and "we'll" becomes "well", but "water-tight" is treated as two separate words.

The operation of VOCLIST is illustrated in Table 4.5, which contains two specimen extracts from its output, showing the most common 26 items from two different text sources.

Table 4.5 -- Specimen Vocabulary Listings.

```
Poet.vox  09/22/94 11:18:50
50212.                    Freq.  Rank  Char  Percent     Cum.%
the                        3497     1     1  6.96447     6.9644
and                        1702     2     2  3.38963     10.354
of                         1287     3     3  2.56313     12.917
a                          1076     4     4  2.14291     15.060
i                           962     5     5  1.91588     16.976
to                          930     6     6  1.85215     18.828
in                          901     7     7  1.79439     20.622
you                         637     8     8  1.26862     21.891
that                        527     9     9  1.04955     22.940
my                          433    10     :  0.86234     23.803
is                          406    11     ;  0.80857     24.611
with                        405    12     <  0.80658     25.418
for                         358    13     =  0.71298     26.131
on                          348    14     >  0.69306     26.824
it                          295    15     ?  0.58751     27.411
me                          272    16     @  0.54170     27.953
but                         272    17     A  0.54170     28.495
not                         265    18     B  0.52776     29.023
your                        247    19     C  0.49191     29.514
as                          247    20     D  0.49191     30.006
all                         240    21     E  0.47797     30.484
be                          233    22     F  0.46403     30.948
or                          228    23     G  0.45407     31.402
from                        226    24     H  0.45009     31.853
have                        222    25     I  0.44213     32.295
he                          217    26     J  0.43217     32.727


Mags.vox  09/22/94 11:21:21
23424.                    Freq.  Rank  Char  Percent     Cum.%
the                        1404     1     1  5.99385     5.9938
of                         1052     2     2  4.49112     10.484
a                           642     3     3  2.74078     13.225
and                         607     4     4  2.59136     15.817
to                          561     5     5  2.39498     18.212
in                          522     6     6  2.22848     20.440
is                          322     7     7  1.37466     21.815
learning                    319     8     8  1.36185     23.177
for                         285     9     9  1.21670     24.393
```

that	250	10	:	1.06728	25.461
this	182	11	;	0.77698	26.238
are	177	12	<	0.75564	26.993
as	170	13	=	0.72575	27.719
on	153	14	>	0.65318	28.372
an	139	15	?	0.59341	28.966
by	138	16	@	0.58914	29.555
with	135	17	A	0.57633	30.131
be	130	18	B	0.55499	30.686
which	117	19	C	0.49949	31.186
from	114	20	D	0.48668	31.672
we	112	21	E	0.47814	32.150
it	106	22	F	0.45253	32.603
or	89	23	G	0.37995	32.983
language	89	24	H	0.37995	33.363
paper	89	25	I	0.37995	33.743
concept	85	26	J	0.36288	34.106

The first of these vocabularies comes from a corpus of 50,212 words of poetry from the electronic anthology, selected by including the two largest files in the database, as it stood on 1st September 1994, of each poet, or the only file of poets who had less than two. (See Appendix D for more details.) It contains contributions from 26 different authors, and thus constitutes an approximation to a general English poetic vocabulary, with few individual peculiarities. As can be seen, the listing extracted consists exclusively of function words. (The three most common content words in this list were `man', `time' and `love', ranked at positions 51, 52 and 56 respectively.)

The second wordlist is from the combined training set of MAGS, i.e. from the journals *Literary and Linguistic Computing* and *Machine Learning*. This exhibits a similar pattern of decrease in frequency, but, unlike the poetic vocabulary, contains several content words in its top 26 entries.

Both lists are derived from a multi-author corpus, and in both cases the most frequent 26 word types account for about a third of the total number of word tokens in the input text.

PRECODE takes the top n items of such a list (where n=77 in all experiments reported here) and encodes a text file by replacing any of these words with a single character -- beginning at ASCII 49 (`1') -- as shown in the column labelled `Char' in Table 4.5.

Words not in the top 77 vocabulary items are coded as one of the following three characters:

`#'	numbers;
`&'	low-frequency words used within the last 20 words of running text
recent context);
`.'	low-frequency words not used in the last 20.

The fullstop (`.') was the commonest character in all the resulting files, in some cases accounting for over half the total number of characters. Note that the line structure of the text is unaltered by this process, i.e. carriage-returns and line feeds are exempt from conversion.

Note also that the same procedure was applied even to SNOB, for consistency, although for a programming language the above definition of a word is clearly suboptimal.

The general poetic vocabulary described above, from 26 authors, was used on problems LIT1, LIT2, LIT3, AGE1, AGE2, AGE3 and AGE4. On all other problems PRECODE used a word list formed from the combined training data for that problem.

4.5.2 Word-Based Results

Once encoded in this manner, the texts were classified by BBTC. Results are summarized in Table 4.6, below. Again n-gram lengths of 1, 2 and 3 were used. To distinguish them from character n-grams these are referred to as words, digraphs and trigraphs respectively in what follows.

Table 4.6 -- Character-Based and Word-Based Classification Results.

	gramsize = 1	gramsize = 2	gramsize = 3
Success Rate (Characters)	71.99	77.31	76.68
Success Rate (Words)	67.27	66.83	56.41
Log-penalty (Characters)	-1.12	-0.88	-0.86
Log-penalty (Words)	-1.01	-1.07	-1.56
Precall (Characters)	0.6927	0.7433	0.7396
Precall (Words)	0.6497	0.6588	0.5644

It can be seen that on all three measures performance was worse after word-based coding than using the original text (except for log-penalty with gramsize=1). The only problem in which this method was clearly superior was ART2, the distinction between two versions of a computer-poetry generator. It also was marginally better on the FEDS problem with gramsize =

2 (57.82% success rate versus 57.57% and -1.31 log-penalty versus -1.38) but hardly enough to make a practical difference.

Clearly this coding procedure discards information, e.g. about punctuation and low-frequency words, and this is not compensated for by the use of more meaningful sub-units.

In a sense these figures provide quite a strong argument for the centrality of the word, despite its notorious difficulty of definition. The coded texts were typically less than 20% as large as the uncoded versions, yet classification performance only declined from about 77% to 67%. In terms of bits per case, the removal of over 500 bytes per line (over 78%) caused a deterioration of about 0.13 bits per decision. This clearly shows that words are information-rich for classification purposes. However, if maximal accuracy is the goal, using words alone is not enough.

One subproblem where digraphs performed better that digrams was AGE4, the chronology of poems by W.B. Yeats. Figure 4.3 shows the results using digraphs on the test SSCs of Yeats, placed in chronological order on the horizontal axis. The vertical axis is the program's estimate that the work is from the older (rather than the younger) Yeats. Points marked A were written before 1915 and points marked B after 1915.

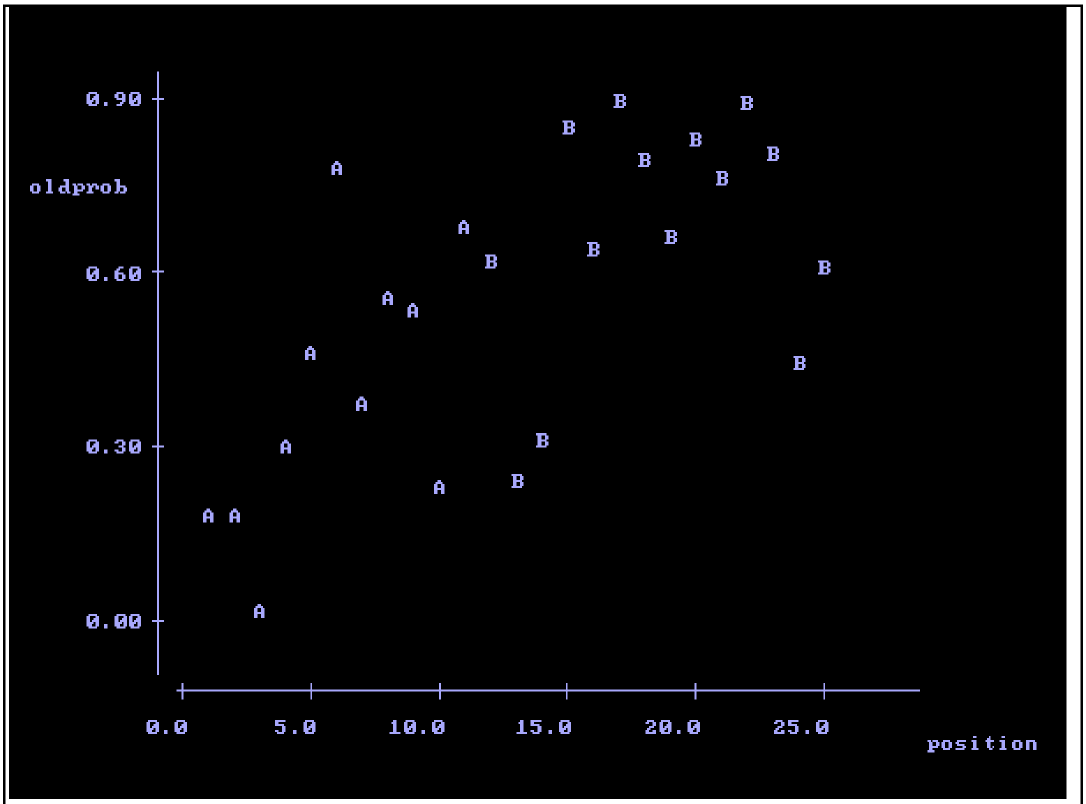


Figure 4.3 -- Early versus Later Yeats.

There is clear upward trend, with a rank correlation of 0.636 between probability of being a later work and ordinal position in the series. The program, trained only on a dichotomy into pre- and post-1915 work, has extrapolated some sort of linear relationship. (However, this method does not work as a general rule, since with early versus later Bob Dylan the same rank correlation was in fact negative.)

4.6 Results Using Word Transitions

For a further attempt to evaluate the word-based approach, two additional programs, SEQ1 and SEQ2, were written to apply Bayesian logic to text classification. These used word-transition probabilities as well as word frequencies, and they did not throw away information by lumping together all low-frequency words.

4.6.1 The Word-Tree Programs

SEQ1 builds a word-transition tree in the following manner. Firstly the top N words are read (where N is a user-supplied parameter) from a VOCLIST output file. Then the successors of each of these head words are found and the most common, for each text source, is recorded. The second most common successor is also recorded, provided it occurs more than nine times. This process constructs a word tree giving word frequencies and the frequencies of certain common word transitions. As an example, Table 4.7 gives the first two entries of such a tree, produced from the Federalist training data (three authors).

Table 4.7 -- Extract from Federalist Word-Tree

```
SEQ1 output;  directory = feds.d1;  date: 09/27/94 17:47:42

C:\TEST\FEDS.HAM  25938
C:\TEST\FEDS.JAY  6052
C:\TEST\FEDS.MAD  24981

the          2340  341  2616
  national     52    8    11
  people       73   28    68
  same         37   13    58
  state        40    3    69
  states       28    3    72
  ##

of           1633  258  1525
  a            99    2    57
  america      11   12    13
  the         536   61   546
```

This table indicates, among other things, that the word `of' occurred 1633 times in 25938 words by Hamilton, 258 times in 6052 words by Jay and 1525 times in 24981 words by Madison. The commonest successor of `of' was `the' for all three authors, but while `the' was about equally likely as a successor of `of' for Hamilton and Madison, `a' was nearly twice as likley a successor for Hamilton as Madison. And `America' (as in `United States of America') was more than six times as likely to follow `of' in Jay's writings as in Hamilton's -- though the absolute frequencies are rather small.

SEQ2 applies Bayes's Rule using such information to provide word and word-transition probabilities. From these it estimates posterior probabilities that each SSC belongs to each possible category.

4.6.2 Summary Results

Table 4.8 gives a summary of classification results on the benchmark suite using SEQ2 with the most common 24 and the most common 48 words as headwords.

Table 4.8 -- Performance of SEQ2 Program.

Score on Unseen Test Cases ↓	24 headwords	48 headwords
Success Rate	66.12%	68.85%
Log-penalty	-1.00	-0.96
Precall	0.6096	0.6382

Overall, these results are less good than those achieved with BBTC using digrams. The only problem where this method gave better results was FEDS with a 61.79% success rate using 24 headwords. Extension to 48 headwords reduced this to 56.33%, however -- presumably by bringing in several words that Mosteller and Wallace would call "dangerously contextual", such as `states' and `power'. (`Upon' came 82nd in the combined word list and hence was not part of either word tree.)

A side-experiment on the FEDS texts was performed by taking the top 100 words in the combined vocabulary and removing all content-words, leaving 75 headwords. A word tree was then built by SEQ1 and passed to SEQ2 which achieved 67.25% success rate on the *Federalist* test cases, giving the confusion matrix below which shows an improvement over the BBTC results using digrams.

Table 4.9 -- Hand-Crafted Federalist Results.

Actual : Predicted ↓	Hamilton	Jay	Madison	Disputed
Predicted Hamilton	206	15	65	101
Predicted Jay	2	3	2	11
Predicted Madison	45	3	62	127

Using this 75-element word-tree SEQ2 assigned more than half the SSCs of the disputed Federalist papers to Madison, whereas, as can be seen from the above figures, it assigned less than a fifth of unseen but genuinely Hamiltonian SSCs to Madison. Once again the disputed papers appear slightly more Madisonian than Madison's undisputed papers, and distinctly unlike Hamilton's undisputed papers.

This illustrates that background knowledge (e.g. about what sorts of words are contextual or not) can be used to improve the performance of a wordtree-based Bayesian classifier. However, it is not directly useful for the present project, whose primary goal is fully automatic (not semi-automatic) text classification.

4.7 Discussion

The main achievement of the work reported in this chapter is the benchmark suite itself (TBS1). This provides an operational yardstick by which the quality of a proposed automatic text-classification method can be assessed. It is far from perfect, but it is much better than having no yardstick at all.

A second achievement is to have demonstrated the feasibility of classifying sonnet-sized chunks (SSCs) of around 640 bytes. This represents a reduction in the minimum length of text to which stylometric techniques can be effectively applied, and may be assumed to have practical consequences.

Thirdly, the tests reported herein give some idea of what constitutes good performance by a text classifier. A baseline success rate of 77% (or an average of 0.88 bits of `surprisal' per classification) on sonnet-sized chunks of text provides a target that any more sophisticated method should ideally aim to surpass.

Fourthly, the experiments carried out in section 4.4 give indirect confirmation of the finding of chapter 3 that short substrings are surprisingly effective stylistic indicators. They also tend to support the contention of Kjell (1994) that letter pairs are useful, at least for initial screening, in text categorization problems.

In addition, the experiments of section 4.5 give some support to the approach pioneered by Burrows (1992), of using high-frequency words as stylistic indicators. Nevertheless, the generally poorer performance of BBTC when information about low-frequency words and punctation was suppressed suggests that high-frequency words alone are insufficient in difficult text classification problems.

Last but not least, the work in the present chapter shows that even simple-minded Bayesianism (ignoring the whole issue of estimating hyperparameters, violating independence assumptions and fudging probabilities based on rare events) can form the basis of an effective text-classification program. This calls into question the reluctance of stylometers, apart from Mosteller and Wallace, to countenance Bayesian methods. The adoption of this Bayesian inferential framework, for all its admitted deficiencies, allowed the fixing of a major problem associated with the cross-codebook compression technique described in chapter 3, namely the lack of a way of quantifying the uncertainty associated with a classification.

Problems remain, however, of which the most important is the fact that neither BBTC nor SEQ1/SEQ2 uses anything that could rightly be called a knowledge base. A word tree does give some hints about differences between text sources, but is much harder to read than, for example, a list of IF/THEN rules. This means that the decisions they make are relatively inscrutable and thus that the classifiers do not yield a readily accessible description of the differences found between the text types under study. This weakness of the semi-crude Bayesian methods described herein prompts a formulation of what is required to make progress from this point, as follows.

Wanted: a method that: (1) achieves 77% success rate or better on the benchmark suite (preferably with over 60% on the FEDS problem); (2) runs as fast as BBTC on a 486-based computer; and (3) employs an intelligible knowledge-representation format, so that the variables it uses, and an outline of their inter-relationships, may be understood without undue effort by a user.

The next chapter will recount an attempt to achieve these goals.

Chapter 5

INSTANCE-BASED TEXT CLASSIFICATION

"Similar memories tend to stick together. The more similar, the greater the `mutual attraction'. Where memory is concerned, the fundamental rule is: *Like attracts like*." -- David Gerlernter (1994).

According to Gerlernter (1994), "thinking is primarily, overwhelmingly *remembering*", and one of the most striking faculties of human memory is its ability to abstract generic concepts from a welter of specific experiences. Ever since Plato's theory of Forms there has been a school of thought that says, openly or by implication, that such concepts are best represented as ideal or prototypical exemplars. For example, Hintzman (1986) argues persuasively that it is only necessary to postulate that traces of particular events are stored in order to explain the human mind's ability to generalize, and supports this contention by providing evidence that abstract concepts can arise from the way in which a pool of specific memories is processed at retrieval time.

No doubt Plato and Hintzman would disagree violently over the fundamental issue of whether ideal forms exist; but they both hold in common the view that knowledge can be represented by prototypes or templates. This chapter describes two classification systems based on that shared view, and summarizes the results obtained by applying them to the benchmark problems described in the last chapter.

5.1 Text Classification by Similarity

A very simple form of learning is rote memory, that is, the storage of previously encountered examples. Rote memory is the basis for what is called case-based reasoning, in which previously solved problems are stored in a case library and novel problems are solved by first matching them to the most `similar' stored problem in that library and then applying (sometimes adapting) the associated solution to the current situation. Currently, case-based reasoning is a growth area in Artificial Intelligence (Kolodner, 1993; Althoff et al., 1995).

Essentially the same idea lies at the heart of the well-tried method of nearest-neighbour classification (NNC) which was first proposed by Fix & Hodges (1951) and has been developed in several directions since then (Dasarathy, 1991). The basic form of this algorithm, single nearest-neighbour classification (1-NNC), works by holding the entire set of training instances in memory: fresh cases are matched to each of these stored instances according to some similarity measure and assigned the class of the case they most closely resemble.

It has been found on numerous past occasions (e.g. Forsyth, 1990; Aha et al., 1991; Fogarty, 1992; Cost & Salzberg, 1993; Michie et al., 1994; McKenzie & Forsyth, 1995) that, although conceptually very simple, this method frequently outperforms more sophisticated classification methods. Thus it was decided to test this technique on the benchmark problems of chapter 4.

5.1.1 Initial Trials

For this purpose two programs were written in C, implementing essentially the 1-NNC algorithm as presented by James (1985). Both programs are the same except that the first (NNC1) uses disc storage for the exemplars, the other (NNC2) reads them into main memory first. The latter runs more quickly but the former can cope with larger data sets. (The results obtained in this section were obtained using NNC2.)

Before applying the program to the 13 text benchmark problems of Chapter 4, some initial trials were conducted on a collection of numeric data sets. The purpose of this preliminary testing was threefold: to confirm that the program worked; to compare two different distance metrics (Euclidean versus City-block); and to assess what level of decline in performance might typically be expected between training and test data when classifying with the nearest-neighbour method.

No strong claims are made about this particular selection of data sets except that they have all been used by published authors in testing various statistical or machine-learning techniques. For completeness, a name and brief description of each data set is given in Table 5.1. Further details may be found in the references attached.

Table 5.1 -- List of Numeric Data Sets.

Data-Set Name & Source	Brief Description	Categories
BANKNOTE (Flury & Riedwyl, 1988)	measurements of images on genuine and counterfeit Swiss bank notes	0=forged; 1=genuine
CARDIAC (Afifi & Azen, 1979)	clinical data on patients admitted to a Los Angeles Hospital with heart failure	1=survived; 2=died
DIGIDAT (Breiman et al., 1984)	quasi-random data simulating a faulty light-emitting diode display, plus four noise variables	numerals: 0 to 9
DOGS (Manly, 1994)	mandible measurements from jaws of five canine species, living and extinct	1=Thai dog; 2=golden jackal; 3=cuon; 4=Indian wolf; 5=prehistoric Thai dog
DRINKERS (Allaway et al., 1988)	blood enzyme measurements of healthy male volunteers obtained by BUPA, plus information about their habitual alcohol consumption	0=light drinker or abstainer; 1=moderate drinker; 2=heavy drinker
FED2 (Mosteller & Wallace, 1984)	frequencies of 22 function words used in extracts from essays by Alexander Hamilton and by James Madison	1=Hamilton; 2=Madison
FISH (Edwards, 1992)	body measurements and biocide concentrations in three species of freshwater fish that inhabit the Tennessee River in Alabama	1=channel catfish; 2=small-mouthed buffalo fish; 3=bass
IRIS (James, 1985)	data on petal and sepal sizes of three species of Iris	1=Iris Setosa; 2=Iris Versicolor; 3=Iris Virginica
QUIN (Quinlan, 1987)	an artificial data set designed to model a task in which only probabilistic classification is possible and which requires a disjunctive concept description	0,1 (assigned by a stochastic rule)
ZOOBASE (Forsyth, 1990)	numeric (mostly binary) attributes describing 101 different animal species grouped into seven zoological classes	1=mammal; 2=bird; 3=reptile; 4=fish; 5=amphibian; 6=insect; 7=other

Except for FED2, these data sets have no specific relevance to stylometry. FED2 is the only concesion to the main theme of this thesis: it was derived from the FEDS data set described in Chapter 4, with the omission of passages by John Jay, by counting frequencies in each line (SSC) of 17 marker words as well as the five commonest words in the combined FEDS text. The 17 markers were just those that appeared in both of two separate lists (one of 31 words, one of 30) used by Mosteller & Wallace as markers in their `main study' and in their `robust Bayesian analysis'. (See Chapter 2 of this thesis for more details.)

Table 5.2 gives further information concerning the sizes of these data sets. (Note that the column giving number of variables excludes the category code itself.)

Table 5.2 -- Details of Numeric Data Sets.

Name	No. of cases	No. of classes	No. of variables
BANKNOTE	206	2	7
CARDIAC	113	2	19
DIGIDAT	1001	10	12
DOGS	77	5	11
DRINKERS	345	3	5
FED2	866	2	23
FISH	146	3	5
IRIS	150	3	4
QUIN	400	2	12
ZOOBASE	101	7	17

Many different distance measures have been proposed for the 1-NNC and its derivatives. Two of the most popular are Euclidean distance (root summed squared deviation) and the `city-block' metric (total absolute deviation). Both these were tried on the above data sets, with the results shown in Table 5.3. Note that a form of jack-knifing was employed, here and subsequently; that is, when used on a single data set, the program finds the nearest neighbour by computing the distance of each case to all **other** instances, excluding the current case itself.

Table 5.3 -- Percentage Success Rate of 1-NNC on 10 Test Problems using Euclidean and City-block Distance.

Data Set	Euclidean Distance	City-block Distance
BANKNOTE	99.51	99.51
CARDIAC	52.21	56.64
DIGIDAT	54.05	54.05
DOGS	83.12	83.12
DRINKERS	40.58	41.74
FED2	61.16	61.57
FISH	74.66	78.77
IRIS	95.33	95.33
QUIN	70.75	70.75
ZOOBASE	98.02	96.04
Mean =	72.94	73.75

A paired t-test on these figures reveals that the success rates of these two distance metrics are not significantly different (t = -1.30, p = 0.23). However, since four of the five cases where the scores differ are favourable to the city-block metric, it was decided to persevere with it as well as the more standard Euclidean metric in the next phase.

The next step involved splitting each data set into two subsets of roughly equal size. Specifically, nine of these 10 data sets were divided randomly into two subsets, each case having a 0.5 probability of being allocated to either. After this approximate halving, the file which in fact contained more cases was designated the `training' or `prototype' file and the other the test file. (The single exception was data-set FED2, where the cases were divided into training and test sets as in subsection 3.3.4.)

Then the 1-NNC program was used to classify both training and test sets, with jack-knifing employed when the training set was used to classify itself, producing the results given in Tables 5.4 and 5.5.

Table 5.4 -- Percentage Success Rate with Split-Half Testing on Numeric Data Sets (Euclidean Metric).

Data-Set [file sizes: training/test]	self-test (jack-knifed)	test on unseen data
BANKNOTE [106/100]	100	99
CARDIAC [62/51]	46.77	56.86
DIGIDAT [507/494]	48.52	52.23
DOGS [39/38]	74.36	76.32
DRINKERS [189/156]	46.03	39.74
FED2 [484/382]	61.16	61.26
FISH [86/60]	68.80	75.00
IRIS [77/73]	93.51	93.15
QUIN [205/195]	72.20	67.69
ZOOBASE [52/49]	92.31	87.76
Mean =	70.37	70.90

Table 5.5 -- Percentage Success Rate with Split-Half Testing on Numeric Data Sets (City-Block Metric).

Data-Set	self-test (jack-knifed)	test on unseen data
BANKNOTE	100	99
CARDIAC	50	70.59
DIGIDAT	49.90	52.83
DOGS	76.92	68.42
DRINKERS	52.38	41.03
FED2	61.57	61.78
FISH	72.09	83.33
IRIS	94.81	89.04
QUIN	72.00	67.69

ZOOBASE	92.31	93.88
Mean =	72.20	72.76

These figures have been presented in full to emphasize one of the most desirable aspects of the 1-NNC technique, namely the fact that it normally gives what Breiman et al. (1984) call `honest' error estimates. That is to say, a self-test on a random sample from a population (provided that jack-knifing is used) will tend to give an error-rate estimate that is not systematically biased towards either under- or over-estimation of the error rate to be expected on another random sample of comparable size from the same population. Remarkably, using either distance metric, exactly half of the data sets show an increase in accuracy between self-test and real test while half show a decrease.

As can be seen by inspection, in neither case are the column means significantly different. A paired t-test between the final column in both these tables also showed no significant difference in success rate between the two distance measures used (t = -0.95, p = 0.37). Finally, it may be of interest to note that although there is an apparent decline in success rate between self test on the full data sets (Table 5.3) and on the unseen part of the split-half data, this is not statistically significant (t = 1.58, p = 0.15) with the Euclidean metric. The difference is even less with the City-block metric.

It was thought important to demonstrate these characteristics of the 1-NNC algorithm on what might be termed `conventional' numeric data sets before dealing with data of more central interest to the present project.

5.1.2 From Text to Numbers

The 1-NNC algorithm, like most statistical pattern-recognition techniques, expects both stored prototypes and test cases to be represented as numeric feature vectors of fixed length. This is emphatically not a natural representation of running text.

In previous chapters of this thesis, all the programs described have worked with text as text, i.e. held as strings of ASCII bytes. To anyone accustomed to computers this seems a relatively `natural' format. For the 1-NNC study, however, this taboo against numeric vector representation was temporarily broken so that texts could be presented to the NNC2 program in suitable format. Accordingly, a Snobol4 program called VARS was written which would transform lines of text (in practice, SSCs of around 640 bytes) into vectors of numbers[13].

[13] This process of transformation from lines of text into feature vectors raises a number of important issues (such as "what exactly constitutes a textual feature?") --

VARS takes two input files as well as the text to be transformed -- one containing a list of strings to be treated simply as substrings, the other containing a list of strings to be treated as words (either file may be empty, but not both) -- and counts their frequencies in each line. These frequencies are written to an output file in a consistent order so that each input line is represented on output by a vector of frequencies. As VARS was applied to SSCs which were of (approximately) equal length, it was not thought necessary to scale these counts by dividing by the line length, although this could quite easily be done. Additionally, the program adds to each output vector two measures of vocabulary richness, the type-token ratio and Brunet's W (Brunet, 1978)[14]. This latter index will have a higher value for a relatively repetitive text than for text with a rich vocabulary. Vocabulary richness measures in general, and Brunet's W in particular, have been found useful in previous stylometric studies of authorship (Kjetsaa, 1979; Holmes, 1992; Holmes & Forsyth, 1995) so the opportunity was taken to include two of them, alongside simple string and word counts, in this part of the analysis.

5.1.3 Trials on Textual Data

The question of what numeric features to derive from text is not, in general, easily answered; but for the present experiment each data set from the benchmark suite was transformed into numeric features in three different ways:

(1) using substrings produced by pairwise chunking as in the cross-codebook experiments (in Chapter 3);
(2) using the 100 commonest words of the composite poetic vocabulary (for verse) or the 100 commonest words in the combined training texts of the group concerned (otherwise);
(3) using the 40 `most distinctive' codebook substrings together with the 40 commonest words.

too many, in fact, to deal with here. Some of these issues will be considered in Chapters 6 and 7.

[14] The formula for Brunet's W is
$W=N^{V^{\alpha}}$**Error! Main Document Only.**
where N is the number of words (tokens) in a section of text, V is the number of distinct vocabulary entries (types) and α is a constant in the range from -0.172 to -0.165. Here α = -0.169 was used. Unlike type-token ratio, W is relatively stable across a wide range of text lengths (N).

Codebook substrings were obtained by creating 100-item codebooks for each training file, as described in Chapter 3, and then combining the codebook entries for each group of texts -- as grouped in the 13 benchmark problems of Chapter 4 -- with duplicate substrings discarded. When it was necessary to rank the most distinctive codebook substrings, the Chi-squared statistic with one degree of freedom was used, an index used successfully for a similar purpose by McMahon et al. (1978). Frequencies were derived from the relevant training sets only.

A program called CHIMARX was written to perform this ranking process. Some sample output from CHIMARX is listed in Table 5.6 which shows the 12 most distinctive codebook substrings from the MAGS training data, i.e. the substrings whose frequency of usage differed most between the two sources (extracts from the journals *Literary & Linguistic Computing* and *Machine Learning*, respectively). The number just after each substring is the Chi-squared score calculated on the basis that the strings would be expected to appear by chance in both classes in proportion to the fraction of the data which came from each class (0.4743 and 0.5257 to four significant digits). The last pair of figures gives the number of occurrences observed in each category. For example, item 10 (the substring `gor', found mainly in `algorithm') appears 15 times in the text from the first source but 118 in text from the second, resulting in a Chi-squared score of over 69. Clearly this is more than merely a chance variation. As might be expected, portions of the words `learning' and `language' appear high in this list. As perhaps might also be expected, there is considerable overlap: `le' and `arn' appear as well as `learning' which contains them, as do `gu' and `angu' alongside `anguag'. This latter issue will be taken up again in section 5.2.

Table 5.6 -- Most Distinctive Substrings from MAGS Data.

```
1  C:\TEST\LL.TRN
2  C:\TEST\ML.TRN
154314 bytes.
proportion in class 1 = 0.474258738
proportion in class 2 = 0.525738022

            Substring  Chi-score     Frequencies
1                 arn  308.561684    12.  380.
2               learn  306.765671    12.  378.
3            learning  250.105076    10.  309.
4           learning   191.725684    9.  241.
5                  le  190.357664    395.  976.
6                  gu  125.015427    182.  30.
7              concep  97.188459     4.  121.
8                 cep  94.2519792    8.  129.
9                prob  72.1900297    27.  149.
```

```
10              gor  69.1053546    15.  118.
11             angu  67.4710513    92.  13.
12           anguag  67.4710513    92.  13.
```

Thus the focus of this part of the study is on the relative merits of common words (advocated by Burrows (1992) among others) versus short substrings as potential stylistic indicators. The numeric data files produced in the three different ways described will be referred to from here on by the labels CB (codebook substrings only), WD (common words only) and BOTH (distinctive codebook substrings together with commonest words)[15].

Firstly, the NNC2 program was tried on the 13 CB data sets, divided into training and test sets as in Chapter 4 for comparability, and the percentage success rates recorded using Euclidean and City-Block distance metrics in two different testing modes (self-test versus test on unseen data). Table 5.7 summarizes the results obtained.

Table 5.7 -- Mean Percentage Success Rates on 13 Text Data Sets, Vectorized using CB Substrings.

	Euclidean Distance	City-Block Distance
Self-test (with jack-knifing)	75.51	77.53
Test on Unseen Data	66.01	66.10

A 2-way analysis of variance of this data revealed a significant main effect of testing mode ($F_{1,36} = 59.22$, $p < 0.0005$) but no significant main effect due to distance metric nor an interaction effect.

Thus, applied to these 13 text benchmark problems, jack-knifed self-testing no longer gives unbiased estimates of the performance of the 1-NNC on unseen cases. A likely explanation for the loss of this rather desirable quality is that whereas nine of the 10 numeric data sets were indeed randomly divided into training and test sets, the textual data was (for the most part) divided according to year of composition. This blocking method was adopted to maintain the integrity of single works, i.e. to ensure that segments of the same work did not appear in both test and training data -- which would have been a kind of `cheating' and would have rendered the benchmark problems unrealistically easy. In addition, the partitioning of the text benchmark data departed further from a 50/50 subdivision than that of the numeric data sets. Thus

[15] Note that the two vocabulary richness indices were included in all three groups of data.

the text benchmark suite constitutes a more severe test than would be expected on the basis of straightforward random splitting.

This interpretation is lent weight by the fact that the only two problems of the 13 to show equal or higher scores on test than training data were ART1 and ART2, the artificial data sets which were indeed random subsets from single sources.

As, once again, no material difference could be found between Euclidean and City-block distance, further trials were conducted using only the more widely-used Euclidean metric.

To complete this section, the nearest-neighbour program was executed on the WD and BOTH data sets, using Euclidean distance. Table 5.8 gives its percentage success rates on test data only, grouped according to the three different ways of vectorizing the benchmark data.

Table 5.8 -- Mean Percentage Success Rate on Text Data Vectorized by three Different Methods.

Vectorization Method	Percentage Success Rate on Unseen Data
CB (codebook substrings only)	66.01
WD (common words only)	61.30
BOTH (distinctive substrings + commonest words)	66.83

A 1-way analysis of variance, controlling for problem number, showed a significant difference ($F_{2,36}$ = 6.73, p = 0.003) between these three levels of accuracy. Clearly vectorization using only the 100 most frequent words gives worse results than the other two methods, involving short substrings. This adds some weight to the suggestion that substrings both shorter than words (e.g. `es ') and longer (e.g. `of the') have merit as textual indicators.

5.1.4 Pros and Cons of 1-NNC

It should be noted, however, that even the best of these three methods (BOTH) gave results markedly inferior to those obtained in Chapter 4 with the basic Bayesian classifier using digrams (66.83% versus 77.31% mean success rate). Thus the single nearest-neighbour algorithm would require modification before it could be put forward as a competitive text classification technique. Overall, 1-NNC, although easy to implement, is revealed by this study to suffer from four main drawbacks:

(1) its accuracy on textual data is lower than the BBTC program of Chapter 4;

(2) it requires all training cases to be stored, thus simulating memorization rather than learning, as that term is usually understood (thus also making the classification phase rather slow);

(3) it uses all features of each feature vector in assessing similarity to memorized cases thus failing to compensate for, or exploit, the manifest redundancy among variables illustrated in Table 5.6, and found in most data sets;

(4) since the `knowledge base' is just the training data, it does not produce an intelligible **description** of what it has learned.

This final deficiency (which, incidentally, this system shares with BBTC) is perhaps the most serious. No cognitive economy is achieved by merely storing all training cases, and thus the human user is not enlightened about the character of the data concerned.

In an effort to alleviate this problem in particular, as well as the other weaknesses listed above, a novel program was developed, based on the 1-NNC method but drastically modified, as described in the next section.

5.2 IOGA : An Instance-Oriented Genetic Algorithm

The need to store all training cases in nearest-neighbour classification has seemed wasteful of both storage space and computing time to previous researchers, and several ways of reducing this wastefulness, while preserving as far as possible the essential simplicity of the basic algorithm, have been devised. Many authors have proposed ways of selecting only a subset of the training cases (Hart, 1968; Swonger, 1972; Ullman, 1974; Gabor, 1975; Ritter et al., 1975; Tomek, 1976; Hand & Batchelor, 1978; Devijer & Kittler, 1982; Fukunaga & Mantock, 1984). Some also have proposed methods that involve the creation of `archetypes', such as centroids, along with or instead of actual training instances (e.g. Batchelor, 1974; Chang, 1974; Batchelor, 1978; Geva & Sitte, 1991): this moves the nearest-neighbour technique away somewhat from pure rote memorization towards true learning in that it usually entails the storage of prototypical instances that were never actually encountered during training; thus it becomes rather like the Learning Vector Quantization, or LVQ, technique of Kohonen (1988). In addition, some authors have sought to avoid using spurious &/or redundant variables by methods of feature weighting &/or selection (e.g. Kelly & Davis, 1991; Smith et al., 1994).

However, to the best knowledge of the present author, no previous researcher has yet proposed a method of economizing on both cases stored and features used at the same time -- the object of the program described in this section.

5.2.1 The IOGA Program

The selection of a suitable subset of variables and cases for nearest-neighbour classification can be seen as an optimization task. It could perhaps be performed in a sequential manner, as in a stepwise regression, but this approach is well known to be vulnerable to interaction effects among variables (MacLachlan, 1992); and in the present case interactions among instances chosen and between variables and instances would also have to be considered. In theory, such an optimization could be performed by exhaustive search, but with a training set containing 326 instances measured on 84 variables (not even the largest tackled in this chapter) that would entail looking at 2^{410} subsets -- clearly not a feasible option. Accordingly, since the genetic algorithm (GA), has been found to be a robust general-purpose optimization technique (Goldberg, 1989), the problem was tackled using an evolutionary method.

IOGA (Instance-Oriented Genetic Algorithm) embodies principles common both to the evolution strategy of Rechenberg (1973) and to the genetic algorithm of Holland (1975). These in turn are based on a biological model, namely the Darwinian idea of evolution by natural selection (Darwin & Wallace, 1858).

In any program of this kind there will be a population of structures representing potential solutions to the problem in hand which can be scored, or at least ranked, by some kind of fitness function. To emulate Darwinian `survival of the fittest', new candidate solution structures are generated by a process analogous to reproduction. Ordinarily this involves a quasi-random selection, biased somehow in favour of higher-scoring members of the population, of two, or sometimes more, parents to which a crossover operation is applied (analogous to mating). A mutation operator is typically applied to the resultant offspring which is then inserted into the population, displacing a low-scoring individual.

Within this general framework there are many variants, differing in details such as how crossover and mutation are implemented. In fact, the two most important attributes of any GA implementation are: (1) the representation scheme; (2) the fitness function used.

In IOGA the representation scheme is quite transparent: each item in the population is a string of R+V bits, where R is the number of records or instances and V is the number of variables in the training data. A 1 anywhere in the first R positions of this bitstring signifies that the corresponding case is to be included among the selected prototypes, a zero means that it is to be excluded. Similarly, a 1 anywhere in the last V positions signifies that the corresponding variable in the feature vector is to be used in distance calculations, while a zero means that it is not. This representation is well suited to being chopped up and recombined by the GA operators.

The fitness of an individual bitstring is computed by running the jack-knifed 1-NNC procedure over the whole training set with only the selected instances used as prototypes and only the selected features employed in distance calculations. The number of correct classifications (K) is recorded during this evaluation. The fitness (F) of that gene-string is then given by

$$F = K - B/(R + V)$$

where B is the number of non-zero bits in the string and R+V, as before, is the total number of bits in the string. Essentially the subtraction of B/(R+V), the proportion of bits used, gives this fitness formula a bias towards brevity which acts as a tie-breaker: for bitstrings with equal error rate the one using less information is preferred. This may be seen as a crude operationalization of Occam's Razor.

Note that K, the number of correct decisions, is summed over all cases in the training set, whether or not they are included by the bitstring in the prototype subset. Note also that, because jack-knifing is used, no case can be its own nearest neighbour.

The version of the GA used in IOGA is novel to this project; but it is loosely based on a procedure called Iterative Genetic Search with Uniform Crossover (IGS-U) devised by Ackley (1987). It can be outlined as follows. (A listing is given in Appendix C.)

1. Create an initial population of random gene-strings, and compute their fitness scores.
2. Pick a parental gene-string at random from the population.
3. Pick a second parent by making P random probes in the population and retaining the gene-string with the highest fitness score (of the P strings sampled).
4. Make P random probes in the population and record the location of the gene-string with the lowest fitness score (out of P sampled).
5. Make a new offspring by applying the uniform crossover routine[16] to the two parental strings.

[16] Uniform crossover makes an offspring by stepping through each string position in turn and at each position picking a binary digit from one or other parent with equal probability.

6. Randomly replace approximately 4% of the bits in the newly created string by random bits (0 or 1 with 0.5 probability). (This will make no difference half the time, by chance, so the effective mutation rate is in fact 2%.)
7. Replace the member of the population selected in step 4 by the newly created gene-string. Also, compute its fitness and if the new string happens to have the best score seen so far, save a copy (outside the gene pool) for subsequent printout.
8. On a proportion of occasions (currently set at a third) apply the mutation routine to a randomly chosen member of the population (and keep a copy of it for later printout if it happens to be the best so far).
9. Increment counters and stop if enough work has been done; otherwise loop back to step 2.

(For all experiments reported in this chapter, P, the number of probes, was set to 4.)

The point to notice about this particular version of the GA is that it is incremental rather than generational. A generational GA consists of a main cycle in which most or all of the population is replaced, by their `descendants', on each step. Generational GAs are more common than incremental ones (Forsyth, 1989; Goldberg, 1989). They are easier to parallelize, but on a serial Von Neumann computer incremental GAs have the virtue of simplicity. The incremental procedure used in IOGA sidesteps certain technical problems connected with fitness scaling, and avoids the expense of sorting as well. In performance there is generally little to choose between these two types of GA (Davis, 1991).

It should be borne in mind that when assessing the amount of work done by a generational GA it is necessary to multiply the number of generations by the population size to give the number of structures tested, while with an incremental GA it is only necessary to count the number of offspring made. Comparison of the number of generations of a generational GA with the number of trials of an incremental GA would be unfair.

Another point worth stressing before presenting the results of IOGA is that no great effort was expended on fine-tuning control parameters such as population size or mutation rate. As Koza (1992), among others, has pointed out, the beauty of genetic algorithms and kindred methods is that they do not depend sensitively upon precisely chosen values of key parameters. This contrasts starkly with some rival techniques, such as simulated annealing or back-propagation, where correct choice of parameters is often crucial (Ritter, 1991).

Nevertheless the above algorithm is not completely untried: a brief preliminary test was conducted on an optimization problem described by Jennison & Sheehan (1993). They found that, although simple, this problem was not very well handled by a `standard' version of Holland's GA as described by Goldberg (1989). Comparison with the results of Jennison and Sheehan indicated that the algorithm described here converged to a solution more than 80 times faster than the standard (Holland-style) genetic algorithm. There are today many variations on the theme of evolutionary computing (see, for example: Back et al., 1991; Fogel, 1993; Kinnear, 1994). The procedure used in IOGA is a relatively non-standard but efficient modern variant.

5.2.2 Results of IOGA on Numeric Data Sets

A program, IOGA, implementing the method described in subsection 5.2.1, was written in C, together with some supporting software including programs called CF and NARC. CF (Compact File) takes the best bitstring dumped by IOGA and creates from it a file containing only those training cases and features needed to make a classification. This is referred to here as an `archetype' file. NARC (Nearest Archetype Classifier) applies the 1-NNC procedure to a full datafile but uses only the instances and features selected by IOGA.

Before testing IOGA on the 13 text benchmark problems, it was applied to the 10 numeric data sets described in 5.1.1, split into training and test sets as in that subsection. Results obtained are given in Table 5.9. These were obtained using NARC after IOGA had been run with a population size of 42 for 1200 trials (**not** 1200 generations, thus quite a short run as GA experiments go). To smooth out random fluctuations, the median value from three runs is quoted. Euclidean distance was employed, so the comparable figures for the standard 1-NNC are in Table 5.4.

Table 5.9 -- Results of Applying IOGA to Numeric Data Sets.

Data Set	jack-knifed self-test (%)	test on unseen data (%)	cases selected	variables used
BANKNOTE	100	98	4	2
CARDIAC	74.19	64.71	4	1
DIGIDAT	72.98	69.23	70	6
DOGS	89.74	84.21	9	3
DRINKERS	63.49	48.72	26	2
FED2	76.03	64.66	80	8

FISH	80.23	75.00	9	1
IRIS	97.40	94.52	6	1
QUIN	74.15	67.18	7	2
ZOOBASE	88.46	79.59	10	4
Mean =	81.67	74.58	22.5	3.10

The figures in the second and third columns are percentage success rates. The last two columns give the number of instances kept by IOGA and the number of variables, or features, selected.

These results show firstly that the `honesty' of jack-knifed self-testing has been lost: all 10 data sets show a decrease from self-test to test on unseen data.

Secondly there has been, as intended, a substantial reduction in size from the full data file to the archetype file, as shown by the number of cases and variables needed for nearest-archetype classification. These figures may be compared with Table 5.2. The best measure of storage required is the product of number of instances multiplied by number of features used. Using this measure the mean size of the archetype files, as a percentage of the storage needed by the full data sets, was 3.45% -- a compression ratio of approximately 28 to 1.

Thus there is indeed an economization of storage, but this would be of no value if it was accompanied by a loss of accuracy. However, the mean success rate of 74.58% obtained here on unseen data is actually higher than that obtained by the basic 1-NNC using the full data set (70.90%, as shown in Table 5.4). A paired (two-tailed) t-test shows that this difference is not significant ($t = 1.67$, $p = 0.13$). Nevertheless, it can be confidently asserted that this genetic subset selection process has incurred no loss of accuracy.

In this connection it is interesting to note that Ritter et al. (1975) compared the performance of three different instance-selection algorithms (condensed, reduced and selective nearest-neighbour classification) on mass-spectrum data and found all three gave slightly worse performance than 1-NNC on the full training data. Likewise, Chang (1974) tested his prototype-based algorithm on some liver disease data and found it had a slightly higher error rate than 1-NNC on unseen data. In other words, it is somewhat unusual to find an instance-selecting version of the 1-NNC that gives better results than the standard algorithm.

Finally it should be mentioned that classification is considerably faster using the archetype subset than the full data. However, the genetic selection process itself is rather slow.

5.2.3 Results with Text Data

For the textual trial IOGA and NARC were applied to the 13 vectorized data sets (see subsection 5.1.3). Of the three vectorization methods (labelled CB, WD and BOTH), BOTH gave the highest success rate on test data with the full 1-NNC and it also had fewest features, 82 active variables plus an identifying field and a category code; so only the BOTH data sets were used in what follows.

IOGA was run with a population size of 42, for only 1024 trials, using Euclidean distance. The results obtained are given in Table 5.10.

Table 5.10 -- IOGA Results on Text Data Sets.

Data Set	jack-knifed self-test	test on unseen data	cases kept / total cases	variables kept
LIT1	69.02	49.09	44 / 326	13
LIT2	75.91	59.32	22 / 137	14
LIT3	77.33	48.91	26 / 172	12
FEDS	69.19	51.61	86 / 542	21
SNOB	89.43	70.92	45 / 246	12
AGE1	91.30	61.73	12 / 92	16
AGE2	100.	90.00	4 / 32	2
AGE3	97.22	63.33	4 / 36	9
AGE4	94.87	64.00	9 / 78	12
MAGS	96.31	82.91	34 / 244	9
WAVE	85.71	61.90	21 / 112	14
ART1	100.	100.	4 / 128	2
ART2	93.97	85.15	10 / 116	9
Mean =	87.71	68.37	24.60 / 173.92	11.15 / 82

Qualitatively speaking, these results tell much the same story as the results obtained with numeric data:

(1) despite jack-knifing, there is a systematic difference between self-test mode and testing on unseen data;
(2) a huge reduction in storage requirement has been effected;
(3) there is no significant decrease in accuracy between the full 1-NNC and NARC on unseen data (paired t = 0.78, p = 0.45);
(4) NARC runs much faster than the full 1-NNC, at the cost of a very slow training phase, especially with the larger data sets.

The reduction in storage needed is even more dramatic here than with the numeric data sets. On average, the archive files took up a mere 1.91% of the space required by the corresponding full data files -- a compression ratio of over 50 to 1. This is achieved by discarding about 87% of training cases and about 86% of the features. To put this in perspective, Chang reported that his method (1974) dropped just over 93% of all training cases of a liver-disease data set, while the most economical selection method of three tested by Ritter et al. (1975) dropped 65.35% of a training set of chemical data. Both of these methods kept all features, as they were purely concerned with selecting reference vectors. Thus IOGA exhibits to a high degree the compression that, as was argued in Chapter 3, is one of the hallmarks of learning. Indeed on this score it is very impressive, surprisingly so in view of the fact that the bias towards brevity in the fitness function was, in essence, only a tie-breaker. (Presumably the presence of spurious variables and rogue examples also creates selection pressure in favour of sparse subsets.)

Nor was this thriftiness in storage bought at the cost of a decline in accuracy. In fact, once again, the mean success rate of 1-NNC using the archetype files was slightly better than with the full data files (68.37% versus 66.83% on unseen data), though this difference is not statistically significant.

5.2.4 Review

The fact that IOGA/NARC gave respectable results in finding archetypal subsets in search spaces ranging from 2^{116} to 2^{626} points after testing a mere 1024 (2^{10}) candidates is, in itself, a vindication of the Darwinian approach to optimization. Once again, evolutionary methods have been shown as robust and effective.

If we consider, for instance, that Koza (1992) uses a population size of 500 and 50 generations (i.e. 25,000 trials) as default values for the genetic programming experiments in his book or that Paechter (1994) describes an evolutionary timetabling system that normally examines 30,000 potential solutions, it will be apparent that 1024 is a very small number of trials in the context of evolutionary computing. Indeed, it is legitimate to ask whether the system has even begun to converge after such a short run; and this query leads on immediately to one of the two main shortcomings of the IOGA program. The figure of 1024 was not chosen to ensure convergence but because anything much bigger would have made the program unacceptably slow. On the FEDS data set, for example, containing 542 cases with 84 numeric features, IOGA took more than five hours to complete 1024 iterations. It took much less time on the smaller data sets, but unfortunately it is precisely with large data sets where data reduction is most useful.

In essence, what IOGA does is exchange the fast training and slow classification normally found with 1-NNC algorithms for the reverse situation.

Some informed commentators (e.g. Openshaw, 1988) have argued, quite sincerely, that the pace of technological development in computing means that computer-intensive methods such as the above should be welcomed; but the present author's opinion is that a program that takes over five hours on a 66-megaherz 486-based computer to perform an unexceptional task ought to be taken as a sign that a more elegant solution must be sought.

Thus slowness must be counted a weakness of IOGA, though it should be noted that this is due primarily to the $O(N^2)$ nature of the 1-NNC algorithm and not intrinsic to the GA itself.

A second major problem with this instance-based approach is the extent to which it provides a user with insight into the structure of the data. As was stated in 5.1.4, this is an important function in machine learning, but traditional nearest-neighbour classification is very poor in this respect. One of the aims of the archetype-selection process was to remedy this weakness.

5.3 From Prototypes to Profiles

For this purpose, an ancillary program called PROFILE was written in C. The function of this program is to take the archetypes selected by IOGA and display them in a semi-pictorial manner, with the intention of illuminating some of the relationships implicit in the data.

PROFILE does this by reading the archetype data and converting each archetype into a string of V integers, where V is the number of variables kept by IOGA, and each integer is calculated by truncating twice the z-score (number of standard deviations from the overall mean) of the present case on the variable concerned. An archetype with the same category code and the same string of truncated z-scores as another is considered to be a duplicate and ignored; hence a further degree of compression (usually small) may be imposed by this program.

A very simple illustration of the results of such profiling on the CARDIAC data is shown below as Table 5.11.

Table 5.11 -- Profile of CARDIAC Archetypes.

```
C:\PC\TA\PROFILE.EXE started on Sat Jan 21 21:26:54 1995
Archetype data from file : \df\hosp.df1
Cases used = 4
Vars used =  1

Variable 6 : `ma_press`
Mean = 74.306452  S.D. = 23.248895

                                value    z-score     Profile
Archetypes of class 1 :
Case 5  562
   6 ma_press       moderate    65.0000  -0.4003           O

Archetypes of class 2 :
Case 50  658
   6 ma_press       moderate    59.0000  -0.6584          oO
```

This is about the simplest imaginable set of archetypes for a 2-class problem. The 18 variables have been reduced to one (mean arterial pressure) and each class is represented by a single archetype. IOGA actually found four, two in each class, but they were two pairs with equal values on the single variable used, so two of them are ignored in this profiling. (Jack-knifing virtually ensures that every category will have at least two archetypes, but, as in this case, they may be equivalent.)

Some insight is provided by the fact that a single most-important feature has been found which gives more accurate classification of unseen data than using all 18 features. It can also be deduced, in a 1-dimensional case like this, that the two prototypes implicitly define a predictive rule

IF mean arterial pressure >= 62
THEN Class 1 (survivor)
ELSE Class 2 (fatality)

although this requires a little background knowledge of how the 1-NNC works.

A more realistic example is illustrated in Table 5.12, derived from the DOGS data (Manly, 1994). Here the three variables retained are as follows: X4, height of mandible below first molar; X5, length of first molar; X7, length from first to third molar inclusive.

Table 5.12 -- Archetype Profiles from DOGS Data.

```
C:\PC\TA\PROFILE.EXE started on Sat Jan 21 21:28:42 1995
Archetypes from data file : \df\dogs.df1
Cases used = 9
Vars used =  3

Variable 4 : `x4`
Mean = 22.358974  S.D. = 3.107607
Variable 5 : `x5`
Mean = 20.948718  S.D. = 2.470337
Variable 7 : `x7`
Mean = 33.051282  S.D. = 4.576535

                                value    z-score     Profile
Archetypes of class 1 :
Case 2  "modern
   4 x4             moderate    22.0000  -0.1155           O
   5 x5             moderate    20.0000  -0.3840           O
   7 x7             moderate    32.0000  -0.2297           O

Archetypes of class 2 :
Case 7  "jackal
   4 x4                  low    17.0000  -1.7245        oooO
   5 x5                  low    18.0000  -1.1937         ooO
   7 x7             moderate    32.0000  -0.2297           O

Case 8  "jackal
   4 x4             very low    16.0000  -2.0463       ooooO
   5 x5             moderate    19.0000  -0.7888          oO
   7 x7             moderate    31.0000  -0.4482           O



Archetypes of class 3 :
Case 16  "cuon
   4 x4             moderate    23.0000   0.2063           X
   5 x5                 high    23.0000   0.8304           Xx
   7 x7             moderate    30.0000  -0.6667          oO

Case 18  "cuon
   4 x4             moderate    24.0000   0.5281           Xx
   5 x5             moderate    21.0000   0.0208           X
   7 x7                  low    29.0000  -0.8852          oO


Archetypes of class 4 :
Case 25  "indwolf
   4 x4             moderate    24.0000   0.5281           Xx
   5 x5                 high    25.0000   1.6400           Xxxx
   7 x7                 high    41.0000   1.7368           Xxxx

Case 32  "indwolf
   4 x4                 high    25.0000   0.8499           Xx
   5 x5                 high    24.0000   1.2352           Xxx
   7 x7                 high    41.0000   1.7368           Xxxx


Archetypes of class 5 :
Case 37  "thaidog
   4 x4             moderate    23.0000   0.2063           X
   5 x5             moderate    19.0000  -0.7888          oO
   7 x7             moderate    32.0000  -0.2297           O

Case 38  "thaidog
   4 x4             moderate    23.0000   0.2063           X
   5 x5                  low    18.0000  -1.1937         ooO
   7 x7             moderate    32.0000  -0.2297           O
```

The profiles themselves appear in the final column, composed of O's and X's. These are derived from the truncated z-scores. The `spine' of the profile is O if the z-score on the variable concerned is negative, otherwise X. Lower case x's or o's extend from the spine rightwards or leftwards, 1 for each half standard deviation away from the mean.

This format was adopted in an attempt to follow in the footsteps of Tukey (1977) who devised several simple but effective semi-graphic ways of presenting data as aids to exploratory data analysis. In the above example, with nine instances and three variables retained, it is arguable that these profiles do provide some insight into the data.

An even simpler, but more relevant, example arises from the text data set AGE2 (poems by Emily Dickinson written up to or after 1863). IOGA picks four cases and two variables as representative of this training data, as illustrated in Table 5.13.

Table 5.13 -- Archetype Profiles for Emily Dickinson.

```
PROFILE.EXE started on Sat Jan 21 21:33:14 1995
Archetypes from data file : \both\age2.df1
Cases used = 4
Vars used =  2


Variable 1 : `--`
Mean = 17.25  S.D. = 9.252724
Variable 6 : `, `
Mean = 4.40625  S.D. = 4.549792

                                value    z-score     Profile
Archetypes of class 1 :
Case 5
   1 --             moderate    18.0000   0.0811            X
   6 ,                  high     9.0000   1.0097            Xxx

Case 13
   1 --             moderate    11.0000  -0.6755           oO
   6 ,             very high    17.0000   2.7680            Xxxxxx


Archetypes of class 2 :
Case 26
   1 --                  low     0.0000  -1.8643         oooO
   6 ,              moderate     6.0000   0.3503            X

Case 32
   1 --             moderate     8.0000  -0.9997           oO
   6 ,              moderate     1.0000  -0.7487           oO
```

IOGA has found a remarkably simple `time stamp' for this author, involving usage of the long dash (represented here as `--') and the comma-space character pair[17]. It is

[17] The dash is virtually an Emily Dickinson trademark (as a rate of over 17 occurrences per 636 characters suggests). In the handful of poems of hers published during her lifetime, however, most of her eccentricities of punctuation were altered by well-meaning editors -- which may have prompted her to use them less often as she grew older.

quite easy to translate this profile into the following brief characterization of the difference between this poet's early and later writings:

Early work:	
dash:	moderate (11..18)
comma-space:	high or very high (9..17)
Later work:	
dash :	low/moderate (0..8)
comma-space :	moderate (1..6)

Bearing in mind that the archetypes from which this description was obtained were 90% successful in classifying unseen samples of her work, this must have something to say about the literary development of Emily Dickinson. It may seem rather trivial, but then discoveries often do -- after the event -- and it is interesting to note that punctuation marks were found by Butler (1979) to be the most important indicators in distinguishing Sylvia Plath's early from late poetry. In other words, this exercise has the virtue of suggesting a research conjecture: that whereas content-free words and short collocations tend to be useful in distinguishing between authors, other characteristics such as punctuation, and perhaps verse form, are needed to distinguish between writings from different ages of the same author.

So profiling has some uses, but AGE2 gave rise to the simplest archetype file among the 13 text data sets. When it comes to a more complex example, such as MAGS, with 34 prototypes measured on nine variables, the PROFILE printout is too long to be of much help in understanding the data. Even here, however, a list of the variables retained does reveal something about this data set.

Substrings:
`arn' `gu' `concep' `prob'
Words:
`_learning' `_language' `_system' `_problem' `_has'

(Words have been given an underscore prefix here as a reminder that their frequencies are not counted in exactly the same way as substrings.)

Moreover, the MAGS archetype file was small enough to be exported into MINITAB (Ryan et al., 1985) and subjected to a Principal Components Analysis (unlike the full data set). The coefficients of the first principal component are listed below in Table 5.14, separated into positive and negative values. They convey a little more information about which variables are most distinctive.

Table 5.14 -- PC1 Coefficients for MAGS Data.

Machine Learning Markers	Coefficients	Literary & Linguistic Computing Markers	Coefficients
`_learning'	-0.5367	`gu'	0.4218
`arn'	-0.5254	`_language'	0.3583
`concep'	-0.2633	`_system'	0.1257
`prob'	-0.1517	`_has'	0.0663
`_problem'	-0.1305		

Again these might seem obvious, with hindsight, although the inclusion of `_has' is hardly predictable, nor perhaps the fact that `_system' is a marker of *Literary & Linguistic Computing* (rather than *Machine Learning*).

This first principal component explains only 28% of the variance among the archetypes, but it does allow perfect separation between the two categories in the archetype file, suggesting that a linear discriminant function derived from the archetypes might be effective on this data. Accordingly, a linear classifier was constructed from this archetype file and tried on the MAGS test data, where it classified 95 cases correctly out of 117. This success rate of 81.2% is relatively good, although it is marginally lower than the 82.91% achieved by NARC on the same data. It suggests that a case could be made for using IOGA as a kind of front-end, to winnow out spurious features and redundant instances before performing more conventional multivariate analyses on an archetypal subset. However this line of attack was pursued no further here, partly because 81.2% is, after all, less than 82.91%, but mainly because the use of standard multivariate methods is quite a well-worn path in stylometry and from a heuristic point of view it seems reasonable that unexpected findings should be more likely as a result of taking a less well trodden route.

5.4 Concluding Comments

To conclude the analysis of these results, the frequency with which IOGA picked variables of the three different types (codebook substrings, vocabulary richness measures and common words) was tallied in order to throw some light on their relative merits. Over 57% of the variables retained were codebook substrings, 42% were words and only once in a total of 145 occurrences was a vocabulary-richness measure used -- the type-token ratio with the LIT3 data. Thus substring frequencies were somewhat over-represented while word frequencies were somewhat under-represented among the variables selected, as there were equal numbers of each to choose from. This evidence is inconclusive, but it does suggest that short substrings deserve further investigation. Vocabulary richness measures, on the other hand, were used at the rate of 1/145 compared with a chance expectation of 1/41; so they failed to emerge as very promising indicators, at least with such short text segments as used here.

To summarize this investigation of instance-based text classification, it may be said that:

- City-block distance does not appear to have a clear advantage over Euclidean distance for this type of work;
- Short substrings are worth serious consideration as textual indicators;
- Archetype selection using an evolutionary method that searches feature space and instance space simultaneously can be highly effective with small and medium-sized data sets;
- On large data sets of high dimensionality (where it would be most useful) genetic archetype selection is likely to prove too slow to be of practical value[18];
- Archetype profiling can be helpful in comprehending multivariate data, though it is no panacea.
- With nearest-neighbour methods there is no generally accepted way to calculate probabilities in order to quantify the uncertainty associated with each categorization, thus using quality measures other than success rate or error rate (e.g. logarithmic penalty) is impracticable.

Finally, however, it must also be said that the 1-NNC method, even when based on selected archetypes, gave lower mean classification success rates on 13 benchmark problems than the basic Bayesian text classifier of Chapter 4 (68.37% versus 77.31%). This motivates a continuation of the search for a better method of adaptive text classification.

[18] In case re-emphasis of this point conveys an impression of incompetence, it should be said in mitigation that other researchers have encountered similar or worse speed problems. For example, Openshaw (1988) states that a GA-based search for a model of some geographic data took "several days on an AMDAHL 5860", while Kelly & Davis (1991) report training times of "the order of" an hour on a Symbolics Lisp machine for a data set containing 210 cases with only two attributes. (And in the neural computing community, overnight and weekend runs are an accepted part of the scenery.)

Chapter 6

TEXT CLASSIFICATION BY RULE INDUCTION

> "From a physical point of view the noises that come out of my mouth are fairly trivial. My jaw flaps open, and I make this racket: out it comes. But although physically it may be trivial, semantically the most remarkable things occur. People say that I made a statement, or asked a question, or gave a command, or order, or explanation; and that what I said was true or false, or interesting, or boring. We attribute all these remarkable properties to noises and marks." -- John Searle (1982).

In this chapter, and from here on, we revert to treating text on its own terms, that is to say, as sequences of characters that represent marks on paper. This approach is more in keeping with the spirit of the present enterprise, and in any case the vectorization methods tried in the previous chapter failed to provide compelling evidence in favour of the practice of transforming texts into numeric vectors. So this chapter compares and contrasts two trainable text classification systems that work with text data held as strings of bytes. The first constitutes, in effect, an automation of the Robust Bayesian Analysis procedure of Mosteller and Wallace (see subsection 2.1.5); the second is based on a classic machine-learning algorithm, namely ID3 (Quinlan, 1982, 1986), modified to deal with textual data.

6.1 A Robust Bayesian Text Classifier

The software described in this section was inspired by the approach adopted by Mosteller & Wallace (1964, 1984) in their robust Bayesian analysis (see Chapter 2). It can be seen as an attempt to automate and formalize that approach to text classification. It consists of four programs, collectively referred to as MAWS (Mosteller And Wallace System):

(1) CHISUBS -- a program for selecting potential textual markers from training data;

(2) MAW1 -- a program for inducing a probabilistic classification rule using a subset of the markers given to it;

(3) MAW2 -- a program for applying a rule produced by MAW1 to unseen test data;

(4) MAWK -- a program for displaying the induced knowledge in a form that helps a human user interpret what has been learnt.

Hence this system treats text categorization as a four-stage process: (1) search for textual features; (2) combine selected features into a classification rule; (3) validate the rule on unseen data; and (4) present the rule for human consumption.

In developing these four programs, an attempt was made to combine (or `hybridize') some of the more successful aspects of the systems already described in Chapters 3 and 4 of this thesis, namely:

- ➢ the effectiveness of short substrings as textual descriptors;
- ➢ the relative success of Bayesian reasoning as an inferential framework;
- ➢ the efficacy of the genetic algorithm as a search technique;
- ➢ the naturalness of treating text as text strings rather than numeric vectors.

6.1.1 A Monte-Carlo Feature-Finder for Textual Data

Many studies of pattern recognition and machine learning begin by presuming that a suitable set of attributes or features has already been found, but in text analysis this presumption is more than usually questionable. It is arguable, for instance, that Mosteller and Wallace brought a good deal of background knowledge to the task of finding text descriptors that would distinguish Hamilton's from Madison's writings, and that once they had discovered reliable verbal markers such as `upon' and `while' the game was almost over. As part of an automated inductive system, it would clearly be desirable for this part of the process to be less dependent on human expertise.

Accordingly, the program CHISUBS was written to find textual markers (short substrings) without any guidance from the user, merely by searching through a given set of training texts. This program is in fact a natural outgrowth of CHIMARX (described in 5.1.3) which was used to rank substrings derived from compression codebooks according to their distinctiveness.

The operation of CHISUBS may be described very simply. Firstly the program repeatedly extracts substrings of length S (where S is a random number between 1 and L) from randomly chosen locations in the training text until N distinct strings have been found. Next the best C of these substrings are retained and printed -- where `best' means having the highest Chi-squared score.

In the experiments reported here, the values for the parameters mentioned above were as follows: L=8, N=3600, C=144. Thus the program sought 3600 different substrings, of from 1 to 8 characters long, and then reduced them to the most discriminating 144[19]. A slight refinement was also introduced, in that when one

[19] Strictly speaking the program only dumps **up to** C substrings, since those with a

substring completely contained another and both had identical scores, the shorter one was ignored. However, although this cuts out some blatant overlaps, it by no means eliminates all redundancy from the marker sets, as the example output from CHISUBS shown in Table 6.1 illustrates. This listing shows the twenty most distinctive substrings found after a search of the training texts of Ezra Pound, T.S. Eliot and W.B. Yeats.

Table 6.1 -- Marker Substrings from LIT3 Data.

```
CHISUBS output;  date: 02/05/95 15:59:48
gramsize = 8
1  C:\TEST\EP.TRN
28072 bytes.
2  C:\TEST\TE.TRN
31919 bytes.
3  C:\TEST\WY.TRN
49617 bytes.
proportion in class 1 = 0.256111524
proportion in class 2 = 0.291209167
proportion in class 3 = 0.452674747

        Substring  Chi-squared   EP   TE   WY

              `"`  519.395006    180.  0.  0.
             `" `  156.39174     55.  0.  0.
            `, "`  115.759107    41.  0.  0.
             `."`  89.6518072    32.  0.  0.
             `s,`  71.9608476    153.  92.  96.
           `" an`  69.3619078    25.  0.  0.
            `hat`  63.1653017    55.  91.  264.
          `that `  62.3034056    38.  58.  203.
           `time`  61.8669424    9.  55.  15.
            `god`  45.269919     24.  1.  5.
             `. `  45.0114726    157.  235.  192.
             ` .`  43.8551554    7.  27.  0.
          `." an`  43.324162     16.  0.  0.
           `tion`  38.3911982    17.  56.  24.
           `love`  37.6142364    4.  3.  48.
```

Chi-squared score less than K, the number of categories, are not kept; furthermore, substrings occurring less than 11 times in the combined training text are also dropped. However, none of the 13 benchmark sets processed by CHISUBS actually gave rise to fewer than 144 markers.

```
           `;`  33.9414823    43.  45.  156.
         ` he`  30.7287863    81.  60.  203.
       ` afte`  29.6633762    5.  28.  8.
          `ma`  28.4227972    45.  38.  140.
          `ae`  27.3496663    17.  1.  5.
          `wh`  27.2313888    47.  100.  191.

[The grave accent (`) is used here as a string delimiter, since
      single and double quotation marks may occur in these text
      markers.]
```

It was considered more important to keep this feature-finding program simple than to expend much effort pruning away redundancies and near-overlaps, since this was exactly the task of the subsequent program in the suite. Nevertheless, even this simple printout provides some interesting information. It can be inferred, for example, from the frequencies for each author, listed as the last three columns, that `"' and `god' are markers of Ezra Pound, `time' and `tion' are Eliot markers and `that ' and `love' are Yeatsian markers. But how best to employ such markers to classify fresh texts remains to be determined.

6.1.2 The Bayesian Rule Generator

The main program in this suite is called MAW1 (standing for Mosteller And Wallace method, part 1). It implements a procedure based on the robust Bayesian analysis of Mosteller & Wallace (1984). It can also be seen as an attempt to retain the accuracy of the basic Bayesian classifier described in Chapter 3, but with a simpler and more transparent knowledge base.

As with the original hand-calculated investigation, the program works by **dichotomizing** rates of occurrence of markers (which are not necessarily words) into high versus low frequencies. Mosteller and Wallace (1984) stated that "variation in paper length is a computational millstone in the main study, and we wish to evade that source of trouble here". To do so, they used papers of nearly equal length. Likewise, MAW1 and MAW2 assume they are dealing with text lines of roughly equal length, which with the benchmark text data used here is in fact the case.

The first thing done by MAW1 is to compute the mean rate of occurrence per line of each of the marker substrings supplied to it. (In the present study these are normally output from CHIMARX or CHISUBS but they could come from any convenient source, including an analyst's intuition.) This mean rate is then used as a threshold, and a 2-by-K contingency table is formed for every marker, where K is the number of text categories, cross-tabulating lines according to which category they belong to and whether the marker occurs at or above the mean rate (high frequency) or below the mean rate (low frequency).

As in Mosteller & Wallace (1984), an attentuation constant[20] is then added to each cell of these tables (justified in the original by the assumption of a prior Beta distribution), which are then converted into conditional-probability tables. An example showing one of these tables, again from the LIT3 data, is shown below as Table 6.2.

Table 6.2 -- Example Conditional-Probability Table.

	Pound	Eliot	Yeats
(\`s,\` < 1.9826)	0.2671	0.4624	0.6723
(\`s,\` >= 1.9826)	0.7329	0.5376	0.3277

Here the mean rate of occurrence of the string \`s,' (letter s followed by a comma) is 1.9826 per line of text, averaged across all three authors. If we term a rate of at least 1.9826 as \`high' (effectively 2 or more occurrences in a line since such counts must be integers) then this table tells us that 73.29% of Ezra Pound's lines have a high rate of this marker while only 32.77% of W.B. Yeats's lines have a high rate. Thus, for instance, P(High | Pound) = 0.7329 and P(Low | Eliot) = 0.4624 for this substring.

From these conditional probabilities of marker status given text category the inverse probabilities, of text category given marker status, can be found using Bayes's Theorem. However for such computations to be valid with more than a handful of markers, an assumption of independence between markers must be made, and it is plain from a cursory inspection of Table 6.1 that this assumption is almost certainly false. So the main task of the MAW1 program is to select a subset of these markers which, when linked to make a probabilistic categorization using Bayes's Rule, give answers that are as accurate as possible on the training data. In other words, the program seeks that subset of markers for which the simplifying assumption of independence is least seriously violated.

This is done using exactly the same version of the genetic algorithm as described in Chapter 5 (subsection 5.1.2). Once again, a simple bit-string representation is used. In this case only variables, not instances, are being selected, so a \`1' signifies that the corresponding marker is to be included in the rule while a \`0' means that it is not used. If a marker is included, its conditional-probability table is used in calculating the

[20] The value of this constant in the experiments described in this chapter was 1.618034.

posterior probabilities of each category (author), otherwise it is ignored in that calculation.

The fitness function employed was

$$\sum_i (\log_2(p_i)) - b$$

where p_i is the computed posterior probability of the correct category for a given case i and the summation is performed over all cases in the training data; and b is the number of non-zero bits in the gene-string, i.e. the number of markers used. This formula always gives a negative result: it is maximized (i.e. made as near to zero as possible) by the genetic algorithm.

The left-hand part of this expression implements the logarithmic scoring rule for probabilistic estimates (Cooke, 1991) already mentioned in Chapter 4 and is essentially equivalent to Mosteller & Wallaces's log-penalty measure except that it is expressed to the base 2. It measures cost in terms of degree of inaccuracy. The right-hand part (b) is also, in effect, expressed to the base 2 (as the number of bits needed to specify the subset of markers used); so it was thought appropriate to subtract it without further scaling from the accuracy component of the fitness function. It measures cost in terms of size, i.e. lack of brevity. Information theory (Shannon & Weaver, 1949) provides a rationale for treating both these components as being measured on a common scale, and weighing them together. (See also: Kapur & Kesavan, 1992.)

To summarize, the MAW1 program uses an evolutionary algorithm to select a combination of markers which maximizes the objective function defined above. It produces as output a list the selected markers, their thresholds and the associated likelihood tables, which are passed on to MAW2 for use in classification. This output list can be seen as a probabilistic decision rule composed of a number of elements each having the general form

```
IF (frequency of `marker' >= threshold) THEN
        $P_j = P_j * R_{j1}$  $(j = 1 .. K)$
ELSE
        $P_j = P_j * R_{j0}$  $(j = 1 .. K)$
```

where K is the number of categories and R_{ji} is a likelihood factor derived from the conditional-probability table associated with the marker concerned. This representation constitutes a rather `flat' knowledge base, i.e. one with only a single type of knowledge on a single level, relating evidence to categories.

MAW2 uses rules of this sort to categorize seen or unseen texts. For assessment purposes, the correct class of each text line must be known.

6.1.3 Results on Text Benchmark Data Sets

To assess this Bayesian method of classification, MAW1 was run for 2048 trials on all 13 text benchmark problems (described in Chapter 3)[21] with a population size of 48. In order to smooth out random fluctuations, the figures quoted below are medians from five repetitions.

The main focus of interest in these tests was the difference in accuracy obtained using codebook substrings (described in Chapters 3 and 5) as opposed to using substrings found by the CHISUBS program described in 6.1.1. Table 6.3 gives the number of variables retained and the percentage accuracy obtained by MAW2 on unseen test data for both methods of feature-finding. Strings derived from compression codebooks are labelled MARX, those found by the CHISUBS program are labelled SUBS. In all results reported below, maximum length allowed for text substrings was nine characters.

[21] It is worth noting that MAW1 runs very much faster than IOGA. The longest run in this phase of testing, on the FEDS data, took less than 25 minutes on a 486-based PC. Thus it took less than 1/13th as long to perform twice as many trials as IOGA on the same data, while using more variables (144 rather than 84).

Table 6.3 -- Variables Retained and Accuracy on Benchmark Test Data.

Data-Set Name	MARX: strings kept	MARX: Success Rate (%)	SUBS: strings kept	SUBS: Success Rate (%)
LIT1	49	68.18	47	70.00
LIT2	38	59.32	30	77.97
LIT3	39	76.09	43	65.22
FEDS	61	54.09	48	60.79
SNOB	36	70.52	34	72.51
AGE1	18	69.14	25	60.49
AGE2	7	73.33	10	86.67
AGE3	9	50.00	11	70.00
AGE4	14	64.00	20	68.00
MAGS	23	80.34	26	81.20
WAVE	31	61.90	26	64.29
ART1	5	100.	5	100.
ART2	5	85.15	20	86.14
Mean =	25.77	70.16	26.54	74.10

As far as accuracy is concerned, the mean success rate using text markers from CHISUBS is higher than that using codebook substrings (74.10% versus 70.16%), although this difference is not statistically significant ((paired, 2-tailed) t = 1.56, p = 0.14). It can be said at least that text markers found directly by CHISUBS are not inferior for this purpose than those found indirectly as by-products of creating a compression codebook. In fact, the mean success rate with CHISUBS markers is the best of any method tested so far apart from BBTC using digrams and trigrams (Chapter 4). In addition, it is the only method to achieve over 60% correct classification of the FEDS test data.

Since this method produces an estimated probability for each category, unlike the nearest-neighbour methods of Chapter 5, its accuracy can also be compared by means of logarithmic scoring (subsection 4.4.4) to that of BBTC using digrams and trigrams. Here the results are less good. Whereas BBTC obtained an average log-penalty score (to the base 2) on the 13 unseen test problems of -0.88 with digrams and -0.86 with trigrams, MAW1/MAW2 obtained mean log-penalties of -1.26 using codebook substrings and -1.19 using CHISUBS markers. Examination of some individual results showed that these mean scores were lowered by a relatively small number of very confident but erroneous classifications. Such occasional extreme misjudgements have little impact on the percentage accuracy but affect the log-penalty score quite severely. This suggests that some degree overfitting of the training data has occurred and hence that the attenuation factor used (subsection 6.1.2) may need further attention.

As far as data compression is concerned, there is a marked reduction in number of features required, indicating that much of the redundancy among markers has been eliminated. The mean number of MARX substrings presented to MAW1 (averaged over the 13 data sets) was 97, of which an average of 25.77, or 26.55%, were retained in the classification rules. The number of CHISUBS markers presented to MAW1 was 144 for all 13 data sets, of which an average of 26.23, or 18.22%, were kept. This latter figure represents a substantial saving, although less drastic than that achieved by IOGA. On the face of it, this is quite a surprising result, since in IOGA the bias of the fitness function towards brevity merely serves as a tie-breaker while in MAW1 the fitness function could override a small loss of accuracy for a larger saving in size. MAW1's fitness function might therefore have been expected to cause greater pressure towards abbreviation, especially as it was run for twice as many trials. However, the information supplied by a single variable in MAW1 (whether or not a substring frequency is over a threshold) is less than that supplied by a single variable in IOGA (an actual numeric value), so their respective feature counts are not strictly comparable.

Altogether, the performance of this system is encouraging. It does not quite achieve the target level set at the end of Chapter 4 -- better than 77% success rate on unseen data over the 13 benchmark problems coupled with at least 60% on the FEDS data -- but it comes closer than any other method tested thus far (74.10% and 60.79%). And it does so while reducing the list of textual markers needed to a manageable number. To put this point in perspective, on the LIT3 data set where MAW1 generated a rule using 35 substrings, the BBTC program used 733 distinct digrams. There is no way that the information implicit in those 733 digrams and their relative frequencies could be made readily comprehensible to a human user.

This raises the question of how well rules of the form produced by MAW1 can themselves be understood.

6.1.4 Descriptive Adequacy

A theme of this thesis is that the purpose of rule induction is not only to obtain effective classification rules but also to lay bare -- if possible -- the relationships between the attributes of objects in the domain of concern and their category membership. To assist in this data-descriptive function, the MAWK program was written to take a rule generated by MAW1 and present it in a form that most people find congenial, as a list of badges ranked in order of importance -- where a `badge' is a feature significantly more common in one particular category that in the other(s) from which it needs to be distinguished, e.g. a word or substring characteristic of a particular author.

MAWK works by reading in the list of substrings, thresholds and conditional probabilities produced by MAW1 and, separately for each category, sorting them into descending order of the effect each would have, if used in isolation, in increasing the posterior probability of the category concerned. (As the markers are, in fact, used independently, this happens to provide a non-distorted insight into how important the markers are in the classification process implemented by MAW2.)

Sample output, from the same run on the LIT3 data used to produce Table 6.1, is shown in Table 6.4.

Table 6.4 -- MAWK Output from LIT3 Classification Rule.

```
Robust Bayesian rulesort of lit3.rsf on Sun Feb 05 16:22:12 1995
Rule from : lit3.fil \spit\ta\lit3.str  Sun Feb 05 16:09:56 1995

Best 7 markers of class 0 :[Ezra Pound]
   1  0.918035    ` "`  >= 0.703488
0.436489 0.969606 0.980082
0.563511 0.030394 0.019918
   2  0.720313    `cia`  >= 0.05814
0.838724 0.969606 0.967773
0.161276 0.030394 0.032227
   3  0.703156    `ae`  >= 0.133721
0.690532 0.950822 0.918533
0.309468 0.049178 0.081467
   4  0.688155    `god`  >= 0.174419
0.711703 0.950822 0.918533
0.288297 0.049178 0.081467
   5  0.55408    `er, `  >= 0.290698
0.605851 0.838117 0.844674
0.394149 0.161883 0.155326
   6  0.542775    ` down th`  >= 0.075581
0.881065 0.932038 0.967773
0.118935 0.067962 0.032227
```

```
   7  0.540401    `, and th`  >= 0.27907
0.627022 0.838117 0.844674
0.372978 0.161883 0.155326


Best 8 markers of class 1 :[T.S. Eliot]
   1  0.749268    `ime f`  >= 0.05814
0.965746 0.838117 0.980082
0.034254 0.161883 0.019918
   2  0.694979    `ong the `  >= 0.093023
0.944576 0.744195 0.943153
0.055424 0.255805 0.056847
   3  0.575937    `. `  >= 3.395349
0.605851  0.23702 0.832365
0.394149  0.76298 0.167635
   4  0.568865    `u a`  >= 0.110465
0.881065 0.800548 0.967773
0.118935 0.199452 0.032227
   5  0.537028    `time`  >= 0.459302
0.817554 0.593921 0.832365
0.182446 0.406079 0.167635
   6  0.51976    `its`  >= 0.30814
0.923405 0.669058 0.770816
0.076595 0.330942 0.229184
   7  0.518868    ` .`  >= 0.197674
0.817554 0.781764 0.980082
0.182446 0.218236 0.019918
   8  0.501919    ` at`  >= 0.540698
0.775213 0.406079 0.635408
0.224787 0.593921 0.364592

Best 7 markers of class 2 :[W.B. Yeats]
   1  0.669073    `nd all t`  >= 0.063953
0.965746 0.969606 0.869294
0.034254 0.030394 0.130706
   2  0.654255    `love`  >= 0.319767
0.881065 0.913254 0.610788
0.118935 0.086746 0.389212
   3  0.63661    ` wher`  >= 0.296512
0.881065 0.875685 0.573859
0.118935 0.124315 0.426141
   4  0.60729    `though`  >= 0.354651
0.902235 0.838117 0.598478
```

```
0.097765 0.161883 0.401522
   5  0.604971    `med`  >= 0.168605
0.902235 0.932038 0.746196
0.097765 0.067962 0.253804
   6  0.558992    ` gre`  >= 0.418605
0.711703 0.913254  0.52462
0.288297 0.086746  0.47538
   7  0.519723    ` hear`  >= 0.337209
0.817554 0.856901 0.647718
0.182446 0.143099 0.352282
```

The point about this ordering is that it is relative. For example `love', Yeats's second best badge, is not actually very common in Yeats's writing: over 61% of his lines (sonnet-sized chunks) do not contain the substring `love' (and hence none of the words, `love', `loved', `loves', `lover', `lovely'). However, `love' is even less common in lines by Ezra Pound, and still less so in lines by T.S. Eliot. Therefore if it is found in a line of text, it makes it more likely that it was written by Yeats and less likely that it was written by the other two. Thus if we started by assuming that all three authors were equally likely, with P(Yeats) = 0.333333, the presence of `love' would increase P(Yeats) to 0.654255. (This is the interpretation of the figure immediately preceding each marker string.)

Similar remarks apply to ` down th' as a badge of Pound's authorship and `its' as a badge of Eliot.

Whether this form of presentation is truly revealing must be left to the reader to judge, but perusing it is certainly less daunting than a list of 733 digrams along with their rates of occurrence for these three authors.

This form of presentation also draws attention to two potential shortcomings of this kind of rule:

(1) the problem of incomplete text fragments (e.g. should `ime f' really be `time for' and should `ong the ' be `among the '?);
(2) reliance on contextual words such as `god', `time' and `love'.

Both these potential weaknesses are even more clearly illustrated by the rule generated from the FEDS data, for which MAWK output is shown as Table 6.5. Here all but the top five badges of each author have been omitted, to save space.

Table 6.5 -- Badges of Hamilton, Jay and Madison.

```
Robust Bayesian rulesort of feds.r2 on Mon Feb 06 17:37:40 1995
Rule from : feds.fil feds.str  Mon Feb 06 17:09:03 1995

Best 5 markers of class 0 :    [Hamilton]
   1  0.849389    `upo`  >= 0.149446
0.698191 0.957247 0.989237
0.301809 0.042753 0.010763
   2  0.763925    `re is`  >= 0.079336
0.839756 0.973577 0.976903
0.160244 0.026423 0.023097
   3  0.759708    `upon t`  >= 0.070111
0.843801 0.957247 0.993348
0.156199 0.042753 0.006652
   4  0.550959    `the na`  >= 0.188192
0.718415 0.842935 0.927568
0.281585 0.157065 0.072432
   5  0.500229    ` a `  >= 2.04059
0.601118 0.908256 0.693228
0.398882 0.091744 0.306772


Best 5 markers of class 1 :    [Jay]
   1  0.869032    `deraci`  >= 0.049815
0.977277 0.695963 0.976903
0.022723 0.304037 0.023097
   2  0.839438    `racies`  >= 0.057196
0.977277 0.695963 0.964569
0.022723 0.304037 0.035431
   3  0.777036    `rica `  >= 0.071956
0.961098 0.712293 0.956347
0.038902 0.287707 0.043653
   4  0.750305    `ional g`  >= 0.075646
0.932785 0.728623 0.976903
0.067215 0.271377 0.023097
   5  0.710278    `thre`  >= 0.066421
0.957053 0.777614 0.952236
0.042947 0.222386 0.047764


Best 5 markers of class 2 :    [Madison]
   1  0.760699    `diciar`  >= 0.070111
0.989411 0.973577 0.882345
```

```
0.010589 0.026423 0.117655
   2  0.645932    ` articl`  >= 0.118081
0.948964 0.957247 0.828899
0.051036 0.042753 0.171101
   3  0.628327    `depar`  >= 0.177122
0.953008 0.940917 0.820676
0.046992 0.059083 0.179324
   4  0.598986    `tutio`  >= 0.402214
0.746728 0.973577 0.582225
0.253272 0.026423 0.417775
   5  0.587102    `the fe`  >= 0.190037
0.876159 0.957247 0.763119
0.123841 0.042753 0.236881
```

Here Hamilton's best marker emerges as `upo', although we know that in fact it should be `upon'. If Jay or Madison wrote on the architecture of suitable congress buildings and discussed the merits of a cupola as opposed to a flat roof, or suddenly took a fancy to the word `stupor', the system would erroneously take that as a sign of Hamiltonian authorship. To give another and perhaps more plausible example, the substring `earning' appears from the MAGS data as the single most important indicator that a text comes from *Machine Learning* rather than from *Literary & Linguistic Computing*. In practice, this does distinguish these two sources very well, but it is unnecessarily vulnerable to confusion: given an economics article on wages and incomes policy, it could well lead to a confident decision that the passage was about machine learning. This shows that some sort of automatic extension procedure is desirable, to make sure that each marker string is as long as justified by the training data (i.e. maximally specific).

While this problem, of inappropriate text-fragment boundaries, could in principle be alleviated by a procedure without knowledge of syntax or semantics, the second problem, of reliance on contextual words and phrases, would be hard to cure by purely lexical processing. It begins to look as if the rather poor performance on the *Federalist* data of this system -- and others relying on such superficial markers -- can be ascribed to this tendency to pick up (fragments of) content words, which characterize subject-matter rather than authorship, such as `deraci' above (doubtless a surrogate for `confederacies') and `diciar' (doubtless standing in for `judiciary'/`judiciaries'). This assertion is backed up by the relatively good performances achieved by all these systems on the MAGS data, where just this sort of tendency is rewarded.

No doubt larger text samples would help. However, to cure this problem fully would entail the loss of some of the more attractive features of the present approach --

namely, simplicity and generality. So for the present this problem will merely be flagged, then put aside.

To conclude this discussion, it can be said in favour of MAWK-type displays that they bring such matters into the open. At the very least, they provide food for thought. It can also be said that in this respect the rule-based approach appears more promising than the instance-based approach tried in the previous chapter. In addition, the MAW1 program runs much faster than IOGA and produces rules that give a higher success rate when classifying unseen text data.

6.2 Text Classification with Discrimination Trees

The method of rule induction used in section 6.1 is somewhat unconventional (though MAW1 does share some important features with the IDIOMS package of Fogarty & Ireson (1994)). By contrast, we next turn to a thoroughly `classic' machine-learning technique, the induction of rules in the form of discrimination trees, which are also known as classification trees or decision trees[22].

One of the earliest algorithms for learning rules from examples was the Concept Learning System (CLS) of Hunt, Marin & Stone (1966). This used the discrimination-tree formalism to represent rules induced from data. Since then the basic idea behind CLS has been developed and refined by numerous authors (Quinlan, 1982, 1986; Breiman et al., 1984; Bratko & Kononenko, 1986; Mingers, 1987; Crawford, 1989; Chou, 1991; Safavian & Landgrebe, 1991; Buntine, 1992; Vadera & Nechab, 1994). Today induction of tree-structured classifiers is the most widely used method of machine learning (McKenzie et al., 1993), for two main reasons: in the first place, discrimination trees are relatively easy to understand, at least when not too large; and secondly, efficient algorithms exist for deriving such trees from example data. In particular, effective ways have been devised of dealing with the main problem that dogged tree induction in its early days -- the tendency of such algorithms to grow over-complex, over-fitted trees when presented with noisy data (Quinlan, 1987, 1993; Mingers, 1989b; Quinlan & Rivest, 1989; Ciampi et al., 1991; Forsyth, 1992; Forsyth et al., 1994).

As indicated above, there are many varieties of tree-growing algorithm but they all work by a stepwise or recursive partitioning of the training data. As a first step, they find the attribute or test which is most discriminatory and divide the data with

[22] The phrase `decision tree' has a completely different meaning in the field of operational research (Goodwin & Wright, 1991) so it will not be used here.

respect to the values of that attribute. Quinlan's influential ID3 program (1982) used a measure of entropy (i.e. surprise) for assessing the discriminatory power of each attribute, but other measures have been suggested (Hart, 1985; Mingers, 1989a) and used successfully in practice. For example, the CART system (standing for Classification And Regression Trees) of Breiman et al. (1984) uses the Gini Index of impurity as a measure of split quality.

Having divided the data into two or more subsets on the basis of the most discriminating attribute, each subset is then partitioned in a similar manner -- unless it contains examples of only one class. This process is repeated until all subsets contain only one kind of data (or until some early-termination rule is satisfied). The end-product is a discrimination tree that can later be used to classify samples never previously encountered. As illustration, Figure 6.1, below, shows in diagrammatic form an example discrimination tree grown from a random subset of 69 instances from the CARDIAC data set (Afifi & Azen, 1979) described in 5.1.1.

[Figure 6.1 -- CARDIAC Data: Discrimination Tree. See last page of this chapter.]

Of the 18 variables available to it, the program has picked three: mean Arterial Pressure (mmHg); U.O. or Urinary Output (ml/hour); and whether or not clinical shock was diagnosed. To classify a new patient with this tree, the test (mean Arterial Pressure > 62) would first be made and then, depending on the answer, either the test (Shock present?) or (U.O. > 15) would be made. Any patient would thus be sorted into one of the four leaf nodes at the foot of this tree. The tallies at such nodes (and at non-terminal nodes) in Figure 6.1 give the number of survivors first, then a colon, then the number of fatalities. Only the rightmost leaf node has a majority of fatalities. Cases arriving at that node would be classed as at maximum risk, while other cases would be classed as likely survivors.

It may be of interest to compare this tree with the prototypes found by IOGA, using a different random subset of the same data, in section 5.3. There the prototypes could be translated into a rule (mean Arterial Pressure >= 62) which is extremely similar to the root node of the tree in Figure 6.1.

6.2.1 The TEXTREE Program

The main concern of this part of the present project, however, is not to deal with numeric data sets, but to extend the methodology of discrimination-tree induction to deal with textual data.

Following CART terminology (Breiman et al., 1984), the four main elements of a tree-growing procedure are:

(1) a set Q of tests or questions in the form "does instance i have property P?" that may be asked of all cases;
(2) a goodness-of-split criterion that can be evaluated for any split at any node in the tree;
(3) a stopping rule to decide when to halt splitting;
(4) a rule for assigning every terminal node to a class when the tree is used in classifying fresh cases.

The TEXTREE program, developed for this phase of the project, may be characterized in terms of these four elements.

Tests : All questions used by TEXTREE are binary. They take the form "is S present in the current case?", where S is one of a list of strings presented to the program as input. Both codebook substrings and markers found by CHISUBS were used as feature sets, so the results in this section are comparable with those of section 6.1.

Since all tests are two-valued, the program grows binary trees. Breiman et al. (1984) have shown that this does not lead to loss of generality, and indeed Cestnik et al. (1987) recommend binarization as a way of guarding against certain types of unbalanced splits; so TEXTREE grows only binary trees, which are computationally easier to handle than multi-branch trees.

Testing only for presence or absence of features does, however, represent a slight restriction in comparison with MAW1, to which frequency counts were available; but since in practice the vast majority (over 90%) of markers kept by MAW1 had thresholds between zero and one (as can be observed in Tables 6.4 and 6.5), and were thus, in effect, binary features, the impact of this limitation is unlikely to be serious.

Split Quality : For its goodness-of-split criterion, TEXTREE uses the Chi-squared statistic, computed from a 2-by-K contingency table, in which the rows correspond to presence or absence of a textual feature and the K columns correspond to the categories. Chi-squared is easy to compute and, since Mingers (1989a) has shown that choice of splitting criterion has little effect on the accuracy of discrimination trees on test data (although it can affect tree size) it was chosen for convenience.

However, a non-standard modification was incorporated to deal with small expected values, because it is well known in statistics (e.g. Siegel & Castellan, 1988) that Chi-squared contributions from cells with an expected frequency of less than 5 are suspect -- although the precise value of this `magic constant' 5 is not universally accepted (Ryan et al., 1985). To mitigate this effect, the score used by TEXTREE as a split-quality measure ignores cells with expected values less than or equal to 4. The practical effect of this amendment is to penalize highly lopsided splits, that is, splits where one row of the table is very `pure' but contains much less than half the total number of cases[23].

Stopping Rule : When deciding that a node in the growing tree (subset in the data) needs no further subdivision, TEXTREE uses a three-part stopping rule:

(1) stop subdividing if all cases at the current node belong to the same class;

(2) stop if the number of cases at the current node is less than Minfreq, where Minfreq is a user-supplied parameter (always equal to 4 in the trials reported here);

(3) stop if no test available actually makes any difference, i.e. if all substrings in the test list are always present (or always absent) in the cases left at the current node.

This final stopping condition does not seem to have been mentioned in the literature, but it can happen and needs to be allowed for; otherwise an endless loop will ensue.

Despite condition (2) this stopping rule is somewhat lax and will typically lead to overgrown trees which need pruning. (The question of tree pruning is dealt with in 6.2.2.)

Terminal Node Category Assignment : The question of the class assigned at a terminal node is unproblematic. As is normal in this area, it is decided in TEXTREE by the plurality rule: the most common class among the training instances at each leaf node determines the category of that node when used in classification. In addition, probabilities are calculated for each category in the obvious way. For example, if a terminal node has 15 instances of class A, 3 of class B and 2 of class C, then P'(A)=15/20=0.75, P'(B)=3/20=0.15, P'(C)=2/20=0.1.

[23] Informal trials with and without this refinement seem to indicate that its use leads to somewhat better balanced trees, with a shallower maximum depth.

6.2.2 Minimum-Entropy Tree Pruning

Straightforward application of the above algorithm will tend to give rise to large and over-complex trees, except with unusually well-behaved data; so a program called TRIM was written to take in TEXTREE output and simplify it. TRIM implements the MinCost pruning method of Forsyth et al. (1994), with minor extensions to deal with more than two categories and to deal with textual data.

MinCost pruning fits into a long tradition in cognitive science that emphasizes information economy (Quastler, 1956; Attneave, 1959; Rissanen, 1987; Wallace & Freeman, 1987; Cheeseman, 1990). In particular, the method is closely related to the SP theory of Wolff (1988, 1991). In Wolff's work on grammatical induction, for instance, learning a grammar is viewed as a problem of finding an economical description of the data: given two grammars of equal size, the one that achieves greater compression when applied to the training text is preferred; likewise, given two grammars capable of making equivalent savings by encoding the training corpus, the smaller is preferred.

MinCost pruning approaches the task of finding a fair-sized tree in a closely analogous manner. The objective is to find the tree that permits greatest compression of the training data relative to its own size. To quantify this objective, a measure of tree quality is needed that takes into account both the complexity of the tree and the savings it achieves in encoding the training data. Information theory (Shannon & Weaver, 1949) is used as a theoretical basis to arrive at such a measure which, for computational convenience, is expressed as a cost to be minimized rather than a payoff to be maximized.

TRIM computes the cost of a tree as the sum of the costs attached to all its leaf nodes:

$$cost(T)=\sum_i cost(node_i)$$

where T stands for a complete tree and the summation is over all leaf nodes. The cost of each leaf node has two components, the cost attributable to the node itself plus the cost of the subset of training cases it classifies (assuming optimal encoding):

$$cost(node_i)=d_i+e_i\times n_i$$

where d_i is the depth of the node concerned, e_i is the average entropy of the outcome data at that node, and n_i is the number of cases at that node.

The depth of a node, i.e. the number of steps down from the root of the tree, provides a good index of its cost because TEXTREE only produces binary trees. Thus to reach a node at depth d implies that d binary choices have been made. This is a once-off cost that is incurred by having that node in the (notional) alphabet of symbols used to encode the training data, and is weighed against any savings that can be achieved by using that node/symbol to compress the data.

The second component of the cost function

$$e_i \times n_i$$

6uses standard information-theoretic principles (Edwards, 1964; Jones, 1979) to calculate the cost, in bits, of an optimal encoding of the categorical outcome data from the frequencies of the instances that reach node$_i$, using the formula for average entropy:

$$e_i = -\sum_j (p_j \times \log_2(p_j))$$

7where p_j is the proportion of instances of class j at the node concerned. (Note that the product of 0 by log(0) is defined as zero and thus that the problem of taking the logarithm of zero is avoided.)

Thus this measure is based on the view that a classification tree is an encoding scheme whose efficiency is assessed in terms of its size and the size of the encoded outcome data. Both the size of the tree and the size of the encoded data are measured in a `common currency', namely information-theoretic bits. To put it another way: MinCost concentrates on **descriptive** economy, on finding the most parsimonious description of the relationship between outcome category and feature values in the training data (within the constraints of the discrimination-tree formalism).

Given a suitable quality measure, tree optimization becomes relatively simple. TRIM reads in an unpruned tree and assigns a static value to every node, including non-terminal nodes. This is calculated from the cost formula above by treating each node as if it were a leaf. Then a dynamic value is assigned to each node by summing the best values of its subnodes (i.e. the lesser of their static or dynamic values) using a recursive procedure. Leaf nodes have no subnodes, so their dynamic values equal their static values.

Next the program scans the tree, seeking non-terminal nodes whose static values are less than or equal to their dynamic values. Such nodes are made into leaf nodes by cutting off their descendant branches, as these subtrees do not improve the quality of the tree as a whole[24]. Finally, the pruned tree is printed out in a suitable form.

The tree produced by this procedure is an optimal abbreviation of the original tree, at least with respect to the cost function employed. It does not follow, however, that it is the optimum tree obtainable from the same data, since TRIM, like most such programs, sticks to pruning and does not attempt to rearrange the order of nodes; nor does it compensate for the fact that TEXTREE, like almost all such progams, does no explicit look-ahead.

The tree shown as Figure 6.1, which has four leaf nodes, is, in fact, a result of applying this procedure to an unpruned tree with eleven leaf nodes.

A textual example, not in diagrammatic form but as printed by TEXTREE, is given below as Table 6.6. This is the unpruned tree, containing 12 leaf nodes, created by the program from the MAGS training data using 144 CHISUBS markers as attributes (of which nine are used). Terminal nodes are distinguished by the presence of numbers, of which the first is the total number of cases arriving at that node. As this is a two-class problem, this total is followed by two category counts (the first tallying cases from *Literary & Linguistic Computing*, the second those from *Machine Learning*). Indentation is used to indicate subtree relationships.

Table 6.6 -- Unpruned Discrimination Tree from MAGS Data.

```
`earn`
  `ngu`
    `lin`
       8    8  0
    ~`lin`
       7    2  5
  ~`ngu`
    `text`
       8    1  7
    ~`text`
       99    0  99
~`earn`
  `ngu`
```

[24] As presently implemented, TRIM will never delete the root node, even if its static value exceeds its dynamic value; so the most discriminating test is always kept. (Only with purely random data does this ever make a difference.)

```
    `ur`
       59    59  0
    ~`ur`
       5    4  1
  ~`ngu`
    `text`
       21    21  0
    ~`text`
      `prob`
        `rs `
           5    2  3
        ~`rs `
           7    0  7
      ~`prob`
        `theor`
           7    3  4
        ~`theor`
          `tu`
             11    11  0
          ~`tu`
             7    5  2
```

As a guide to reading this output, it should be noted that the first line of numbers (8 8 0) follows three indented substrings, `earn', `ngu' and `lin'. This can be interpreted as showing that, in this training sample, there were 8 cases containing all three of these substrings and that they all came from *Literary & Linguistic Computing*. A tilde (~) in front of a string denotes negation, so the second line of numbers (7 2 5) can be read as showing that, of the seven instances in the training data where `earn' and `ngu' were present but which did not have `lin', two were from *Literary & Linguistic Computing* and five from *Machine Learning*.

Although most of the substrings used here make sense (`earn' from `learning', `ngu' from `language' and `linguistic', `text' from `text', `texts' or `textual', and so on) even a tree of this comparatively modest size presents some difficulties of interpretation. After pruning by TRIM, however, the picture is considerably clearer -- as shown in Table 6.7 by the trimmed version of the same tree, which uses only five variables and has only seven leaf nodes.

Table 6.7 -- Pruned Tree from MAGS Data.

```
`earn`
  `ngu`
    `lin`
       8   8  0
    ~`lin`
       7   2  5
  ~`ngu`
     107   1  106
~`earn`
  `ngu`
     64   63  1
  ~`ngu`
    `text`
       21   21  0
    ~`text`
      `prob`
         12   2  10
      ~`prob`
         25   19  6
```

It will be seen that some of the deeper leaf nodes, covering few cases, have been amalgamated.

Normally it is the larger subsets which have most to tell us about the structure of the training data. In this case the four largest terminal subsets (containing 107, 64, 21 and 25 instances) cover over 88% of the training sample. Expressing them as conjunctive clauses gives a good summary of what the program has found:

`earn' AND NOT `ngu'

==> 106 to 1 in favour of class 2;

NOT `earn' AND `ngu'

==> 63 to 1 in favour of class 1;

NOT `earn' AND NOT `ngu' AND `text'

==> 21 to 0 in favour of class 1;

NOT `earn' AND NOT `ngu' AND NOT `text' AND NOT `prob'

==> 19 to 6 in favour of class 1.

In terms of badges it is not hard to deduce from this that `ngu' and `text' are badges of class 1, *Literary & Linguistic Computing*, while `earn' and `prob' (found in `problem' as well as `probable' and its derivatives) are badges of class 2, *Machine Learning*. As it happens, the journals' titles themselves are correctly badged on this basis.

A human analyst might be wary about the last conjunctive condition above, which covers cases where all the system's favourite markers are absent. This mixed subset could almost be seen as a `dustbin' grouping -- of atypical cases which failed to find their natural home -- although the software has no such knowledge and would thus assign cases satisfying this last conjunction to class 1 by the plurality rule.

6.2.3 Results on Benchmark Data

To test this tree-simplification procedure and, in particular, to test whether tree simplification can be achieved without loss of accuracy, a program called USETREE was written to take trees produced by TEXTREE and use them to classify seen or unseen texts. It was used on all 13 text benchmark problems. The figures quoted below give both tree size in terms of leaf nodes and accuracy in terms of percentage success rate on unseen data.

Table 6.8 shows the results using codebook substrings (MARX) both with and without post-pruning by TRIM. Table 6.9 gives the same results using CHISUBS substrings (SUBS). To keep a `level playing field', exactly the same sets of MARX strings and SUBS strings were used as in section 6.1.

Table 6.8 -- Discrimination-Tree Results using MARX strings.

	Original Tree		Tree Pruned by TRIM	
Data-Set Name	Leaf Nodes	Success Rate (%)	Leaf Nodes	Success Rate (%)
LIT1	28	60.91	19	60.91
LIT2	16	49.15	9	49.15
LIT3	17	58.70	11	57.61
FEDS	68	50.12	19	52.61
SNOB	14	68.13	9	69.32
AGE1	13	62.96	5	61.73
AGE2	5	63.33	3	63.33
AGE3	5	56.67	4	56.67
AGE4	8	24.00	6	24.00
MAGS	12	85.47	9	85.47
WAVE	13	66.67	6	66.67.
ART1	2	100.	2	100.
ART2	10	71.29	3	75.25
Mean =	16.23	62.88	8.08	63.29

Table 6.9 -- Discrimination-Tree Results using SUBS strings.

	Original Tree		Tree Pruned by TRIM	
Data-Set Name	Leaf Nodes	Success Rate (%)	Leaf Nodes	Success Rate (%)
LIT1	25	57.27	19	57.27
LIT2	15	67.80	10	67.80
LIT3	16	52.17	12	52.17
FEDS	50	58.06	18	57.32
SNOB	13	70.92	10	70.92
AGE1	10	62.96	8	65.43
AGE2	3	73.33	3	73.33
AGE3	6	66.67	3	66.67
AGE4	8	52.00	7	52.00
MAGS	12	83.76	7	83.76
WAVE	11	47.62	9	42.86
ART1	2	99.17	2	99.17
ART2	7	79.21	6	79.21
Mean =	13.69	67.00	8.77	66.76

These figures show clearly that the trimming process is effective. Combining both tables, the total number of leaf nodes in the pruned trees is 56.32% of those in the unpruned trees. This compression effect is greater, the larger the original tree: taking only those cases where the original tree had more than nine leaf nodes, the mean relative size of the trimmed trees was 46.64% of the untrimmed trees. Furthermore, this substantial simplification does not entail a loss of accuracy. The mean success rates, on unseen data, for both full and pruned trees are virtually identical. In most problems, the success rate was the same before and after pruning. Of the eight cases out of 26 where there was a change, four went up and four went down.

This finding echoes a number of similar studies (Clark & Niblett, 1987; Mingers, 1989b; Michie et al., 1994) where various methods of tree pruning have been found generally to have either no effect or a slight beneficial effect on classification accuracy. It confirms earlier studies by the present author (Forsyth, 1992; Forsyth et al., 1994) by extending the application of this particular information-theoretic method of tree pruning to a new domain, text classification, and obtaining broadly the same results as with numeric data sets. It thus also lends indirect support to the SP theory of Wolff (1991, 1993) from which the present approach to tree simplification drew inspiration.

The above figures also bear on the question of whether codebook substrings or CHISUBS markers provide better attributes for the tree-growing process. Comparing the results using pruned trees, eight out of thirteen problems showed higher accuracy with trees built from SUBS strings than MARX strings. A paired t-test of these figures showed a mean difference of 3.476% in favour of the SUBS strings. This is not statistically significant (t = -1.01, p = 0.33) but it is in the same direction as the corresponding comparison using the MAW1/MAW2 software (subsection 6.1.3). There is thus no cogent reason to prefer codebook substrings to text markers found by the CHISUBS program in this sort of work.

6.2.4 Comparision of Tree-Based with Robust Bayesian Results

However, while these results suggest that MinCost pruning is a viable method of tree simplification and that substrings found by a Monte Carlo process deserve further attention as textual markers, they do not show the practice of induction by tree-growing in a very good light.

In terms of success rate, the mean values obtained here using pruned discrimination trees were worse than those obtained with the Robust Bayesian software described in section 6.1. With the MARX substrings the mean difference is 6.872% (paired, 2-tailed t = 2.0, p = 0.068). With the SUBS substrings the mean difference is 8.152% (paired, 2-tailed t = 3.429, p = 0.056). It does not seem theoretically objectionable to combine these paired comparisons and, if this is done (giving 26 differences), the mean difference in success rate is 7.512% (t = 3.65, p = 0.0012) -- leading to a rejection of the null hypothesis that this mean difference is zero.

Of these 26 comparisons, in 20 cases MAW1/MAW2 gave better accuracy, in 5 cases TEXTREE/USETREE gave better accuracy and in one case there was no difference. Using log-penalty as a quality measure, again on unseen test data, would give a less clear-cut picture (16 cases favourable to the Robust Bayesian software and 10 favourable to the classification-tree system) but this is unlikely to be fundamental: it reflects the fact that MAW2's probability estimates tend to be exaggerated. This exaggeration, while it means that these computed probabilities cannot be taken at face value, could probably be cured by applying a `squashing function' in MAW2 to prohibit extreme estimates.

It is worth asking why Bayesian methods, both naive and robust, work better on this sort of data than tree-based induction. There is no *a priori* reason for either method to dominate the other on all data sets, so it must be presumed that the weaknesses of probabilistic Bayesian classification are less brutally exposed by these tasks than the weaknesses of tree-structured classification. Whereas Bayesian methods, of the type used here, are vulnerable to interdependencies among features, tree-structured classifiers are particularly vulnerable to noise or random variability. Quinlan's original application for ID3 was classifying chess endgames as won, lost or drawn. Chess is an abstract problem domain where the attributes are discrete and deterministic classification is possible. While the basic method has been modified by Quinlan and others in many ways (of which tree pruning is an example) to cope better with noisy data, it remains at root a logical rather than stochastic method. Bayesian inference, by contrast, was designed with uncertainty in mind from the outset. When a problem requires aggregation of several items of evidence, each of which has only a weak probabilistic link with category membership, a Bayesian classifier will tend to do better than a tree-structured classifier -- provided that interaction effects between the items of evidence are not very pronounced.

From this it is reasonable to conclude either that interaction effects among the textual attributes used here are relatively minor or, more likely, that the more serious ones have been screened out by MAW1's genetic search for mutually supportive combinations, whilst the problem of random variation is quite severe.

6.3 The Characterization of Character Strings

The mean percentage success rate achieved by the tree-based classifier of section 6.2 is undoubtedly disappointing. In fact, this system proved worse, on this measure, than all other methods benchmarked thus far -- apart from BBTC using trigraphs (subsection 4.5.2). Despite this poor result, the experiment of building a text-oriented tree-induction system has not been worthless, for several reasons.

In the first place, because tree induction is such as well tried method, its use provides a link between the present study and the wider field of machine learning. In a sense, using a tree-based classifier (like using 1-NNC, another standard method) is a way of benchmarking the benchmarks. Indeed, it would be legitimate to take the fact that tree-based classification performs relatively poorly on the 13 benchmark problems as evidence that classifying 640-byte strings is rather a hard task -- and therefore that better-performing methods, such as the Robust Bayesian system of section 6.1, are attaining a high standard of accuracy.

Second, the pruned discrimination trees are actually successful in at least one respect: they are compact. In every one of the 13 test cases the discrimination tree uses fewer variables than the corresponding Robust Bayesian rule.

Third, and most important, a discrimination tree like that in Table 6.7 is an excellent form of data description. As an exploratory mode of analysis, the procedure of finding features with CHISUBS, growing a tree with TEXTREE and pruning it with TRIM provides a perspective on the text that is simply not available in any other way. In other words, the tree formalism is the best language for differential text description of those tried in this project so far.

Both the glories and the failings of discrimination trees in this regard are well exemplified by a tree grown not as part of the benchmarking exercise but during early testing, out of personal curiosity. Figure 6.2 shows a tree produced by TEXTREE for discriminating between writings by my mother (HF) and my father (JF). In this diagram, counts at leaf nodes for HF precede those for JF. This tree uses only four substrings and contains only five leaf nodes, but it correctly classified 33 out of 37 unseen cases (89.19%). It is concise, precise and perspicuous.

[Figure 6.2 -- 'Family' Tree: HF versus JF. See last page of this chapter.]

Here over 91% of the training cases fall into just 2 leaf nodes, so it is not difficult to express the conditions covering the most common subgroups as the following pair of rules:

```
IF it contains a semicolon
        THEN it is by JF  (odds about 39 to 2 on).

IF it does NOT have a semicolon
AND it does NOT have `ing the '
AND it does NOT have `eace'  [= `peace']
        THEN it is by HF  (odds about 39 to 2 on).
```

Brief as it is, this tree combines into a compact stylistic description a wide range of indicator types -- wider than it has been customary to use together in the field of stylometry hitherto:

(1) a punctuation mark (the semicolon, a paternal favourite);

(2) a collocation (not of two successive words or lemmata, but of a grammatical morpheme, `ing', followed by a function word, `the');
(3) a suffix (`k ', which will pick up words ending in `k', such as pick, kick, bark and dark -- not a very good example, though it does show that affixes may be used);
(4) a content word (`eace' in reality must signify `peace' as all occurrences of `eace' in this sample of my father's writings are, in fact, also occurrences of the word `peace' or its cognates `peaceful(ly)' or `peace-maker(s)').

Of course, there are objections to each of these. Punctuation, it is well known, is susceptible to tampering by editors or compositors. The so-called suffix `k ' is admittedly a motley disjunction (look, took, seek, look, back, lack etc.) and in any case only splits five instances into two groups, so is unlikely to be reliable. And the improper fragmentation of the word `peace' could be misleading unless a user somehow puts in the missing initial `p' (prompting the awkward question: how long is a string of `peace'?).

Yet the very fact of making these criticisms affirms the value of this type of representation. It is hard to imagine doing the same with the connection weights leading to a group of processing units in a multi-layer Perceptron. There are no hidden layers here.

The main point is that this system has fewer preconceptions about how stylistic differentiae should be represented than any previous stylometric classifier. It uses a way of describing linguistic habits in which features are not pre-specified and the logical relations between them are discovered rather than imposed. In virtue of this flexibility, trees such as that above may be seen as pointing towards a new style of stylometry. In short, Figure 6.2 is an explicit, if primitive, example of a stylogram -- a species of entity that has heretofore lurked in the undergrowth of stylometry but which has never before been exhibited so plainly (as discussed in Chapter 2 of this thesis).

6.4 Review and Summary

This chapter reports on two different inductive approaches to rule-based text classification. Section 6.1 describes MAWS, which modifies and automates, with the aid of an evolutionary algorithm, what was originally a slide-rule-and-paper technique, the Robust Bayesian analysis of Mosteller & Wallace (1984). Section 6.2 extends the range of application of a well-known method of machine learning

(Quinlan, 1982; Breiman et al., 1984) to textual data. Results obtained in these studies indicate that:

(1) minimum-entropy tree-pruning is a viable method not just with trees derived from numeric data sets but also with those grown from textual data;

(2) Monte-Carlo feature-finding is a promising method of discovering textual descriptors, so long as the problem of inappropriate segment boundaries can be solved.

Further to the second of these points: results in this chapter show that CHISUBS strings are at least as effective as codebook substrings, while results in the last chapter show that codebook substrings are at least as good as frequently used words, which have held pride of place as textual markers in most stylometric studies up till now. As the kind of substrings found by CHISUBS can include a wider range of textual patterns than just words or word-pairs, there is some reason for preferring them.

The results obtained in this chapter also provide further support for the use of a Bayesian inferential framework and for the use of genetic algorithms in combinatorial search (thus reinforcing the conclusions reached in Chapter 5).

Finally, while the Robust Bayesian method was clearly more accurate on the 13 text benchmark problems than the tree-induction system, the latter was superior in respect of data description (and both were better for this purpose than the methods employed in Chapter 5). The next two chapters will recount an attempt to combine the strengths of both these approaches.

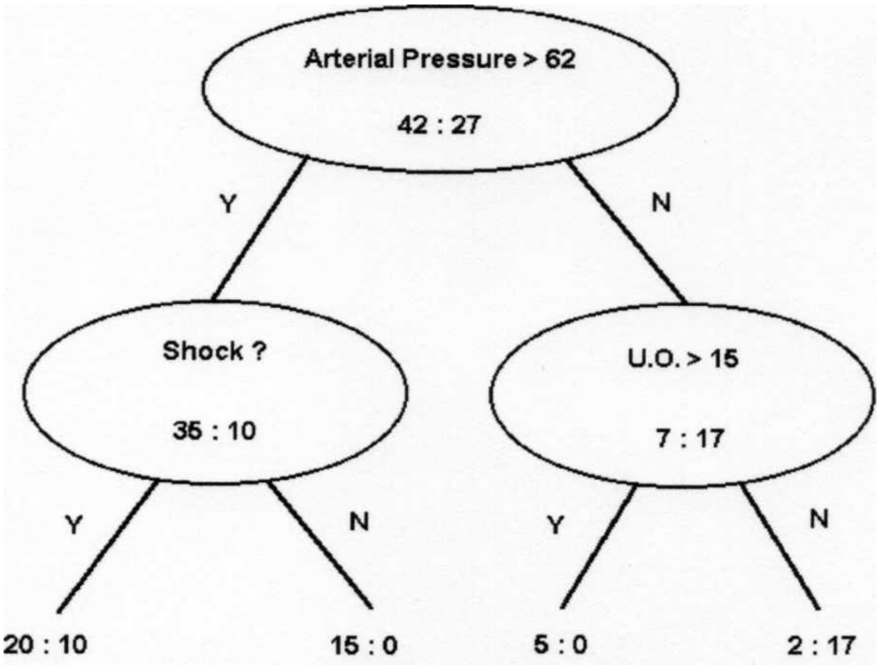


Figure 6.1 -- CARDIAC Data: Discrimination Tree.

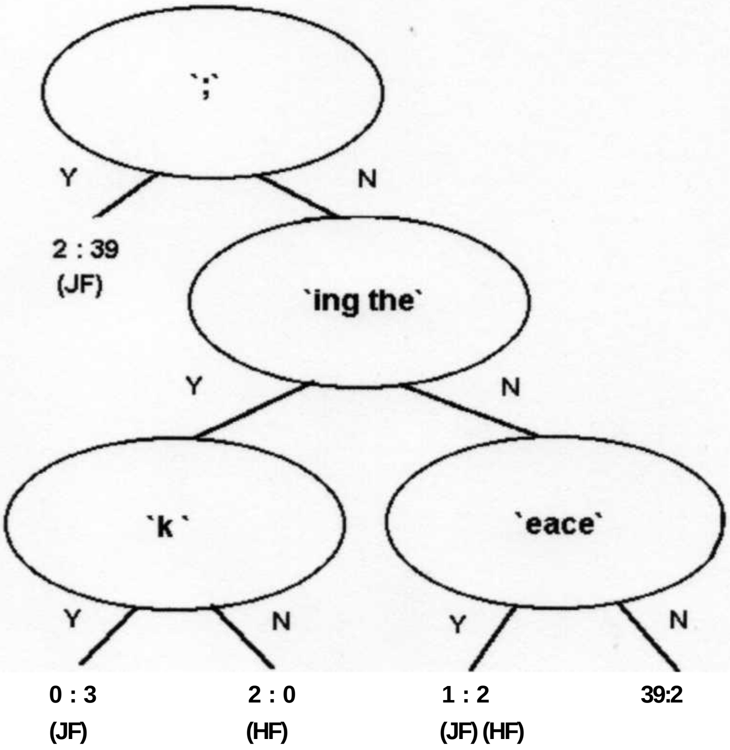


Figure 6.1 -- 'Family' Tree (HF versus JF).

Chapter 7

EXTENDING TEXTUAL FRAGMENTS TO THEIR `NATURAL' LENGTHS

> "Every string which has one end also has another end. -- Finagle's First Fundamental Finding." -- Robert Anton Wilson (1982).

Of the systems described in the last chapter, the robust Bayesian text classifier achieved higher success rates on unseen data but the system based on discrimination trees gave rise to more intelligible descriptions of what it had learned. The next chapter will describe and evaluate an attempt to build on the work of Chapter 6 by combining the best elements of both those systems -- using genetic search and Bayesian reasoning as in MAW1 while generating intelligible descriptions as in TEXTREE.

First, however, some attention is devoted to mitigating the main problem with using text fragments as stylistic markers (described in section 6.3), namely the propensity of the Monte-Carlo feature-finding process to break substrings at `unnatural' points.

7.1 How Long is a Piece of Substring?

As explained in 6.1.1, The Monte-Carlo feature-finding program, CHISUBS, has no background knowledge. It knows nothing about words, morphemes, punctuation, parts of speech or anything specific to English or other languages. It treats text simply as a sequence of bytes. This lack of preconceptions is an advantage in that it could deal with other natural languages such as Latin, artificial languages such as C++, or indeed non-linguistic material such as coded protein sequences, without amendment. But it has the disadvantage that the program often produces substring markers which, to a user, appear to be truncated at inappropriate places. Examples of this problem are `eace' instead of `peace' from the LIT2 data, `rpus' instead of `corpus' from the MAGS data, and, most irritating of all, `upo' instead of ` upon' from the *Federalist* samples. Rules containing such incomplete substrings are less than ideal as descriptions.

Two programs were written to alleviate, though not completely solve, this problem -- without the need to introduce any background knowledge such as a lexicon or morphological rules (specific to English) that would reduce the generality of the method. The first of these, STRETCH, takes a list of substring markers produced by CHISUBS and extends each of them as much as is compatible with the training sample. It is thus a post-processor for CHISUBS. The second program, TEFF (Textual Extended Feature Finder), is a substitute for CHISUBS. It picks short substrings at random by the same method as CHISUBS but each substring is `stretched' as much as compatible with the data as soon as it is generated and before being saved for evaluation. (A listing of TEFF is given in Appendix A.)

The idea behind both programs is that if a substring is embedded in a longer string that has exactly the same occurrence profile then retaining the shorter substring is an inadvertent and probably unwarranted generalization. For example, if `adver' happens always to be part of `advertise' or `advertising' or `advertisement' in every occurrence in a particular sample of text it seems a safer assumption that `advertis' characterizes that text than `adver', which could also appear in `adverbial' or `adverse' or `animadversion' or `inadvertent' -- which, with our knowledge of English, we suspect to characterize rather different kinds of writing.

So both STRETCH and TEFF employ a procedure that takes each proposed marker string and tacks onto it character sequences that always precede and/or follow it in the training text. The heart of this process is a routine called Textend(S) that takes a proposed substring S and extends it a both ends if possible. An outline of its operation is given as pseudocode below.

```
REPEAT

    IF  S is invariably[25] preceded by the same character C
    THEN  S = concatenate(C,S)

    IF  S is invariably followed by the same character C
    THEN  S = concatenate(S,C)

UNTIL  S reaches maximum size or S is unchanged during loop
```

In TEFF, this procedure is only used within the same category of text that the substring is found in. For example, with the Federalist data, if `upo' was found in the Hamilton sample, as it most probably would be, then a common predecessor/successor would only be sought within the Hamilton sample. STRETCH, on the other hand, does not have information indicating where each

[25] Originally `invariably preceded' (or followed) meant exactly that, but the process was rather slow, so current versions of this procedure actually stop looking after 40 consecutive occurrences of the same predecessor or successor. This does not affect substrings that occur less than 40 times, of course; and appears to make little difference to the rest.

substring came from, so it tries each text class separately. Sometimes this leads to different extensions of the same substring. As rare and/or undiscriminating extended substrings are later dropped, this does not necessarily constitute a practical problem: if more than one extension leads to a good discriminator, there is no need to prune them away.

This is a simple but effective procedure which does seem to eliminate the most glaring examples of improper text fragmentation. As this is a rather subjective judgement, a complete specimen of an application of STRETCH to a list of 144 substrings produced by CHISUBS from the *Federalist* data is given below. This shows each input substring, then a colon, then the resultant extended version of that substring -- both bounded by grave accents to show whether blanks are present before or after. Thus,

```
6  `deraci` : ` confederacies`
```

means that the sixth item worth keeping was derived from the substring `deraci' which turned out always to be embedded within the longer string ` confederacies' (in the text from Jay, as it happens). In addition, at the end of this listing, the best 33 extended markers are shown, in order, in a format that gives the marker string itself, then its Chi-score, followed by the frequencies of occurrence in the three authors concerned. From this it hoped that the reader will be able to appreciate how the program works and judge its effectiveness.

Table 7.1 -- `Stretched' Substrings : Specimen from Federalist Text.

```
STRETCH output;  date: 06/27/95 11:32:22
gramsize = 16
input markers from file : \pc\ta\feds.str
0  ` the ` : ` the `
1  `nd ` : `nd `
2  `and ` : ` and `
3  `and ` : `and `
4  ` nation` : ` nation`
5  `our ` : `our `
6  `deraci` : ` confederacies`
7  `deraci` : ` confederacies, `
8  `upo` : ` upon `
9  ` upon ` : ` upon any whom i`
10  `artment` : ` department`
11  `ional g` : ` national govern`
12  `tive` : `tive`
13  `racies` : `racies`
```

```
14  `depar` : ` depart`
15  ` stat` : ` stat`
16  ` stat` : ` state`
17  `slative` : ` legislative`
18  ` power` : ` power`
19  `federac` : ` confederac`
20  `rica ` : ` america `
21  `rs o` : `rs o`
22  `rs o` : `ers of `
23  `rs o` : `rs of `
24  `gislat` : ` legislat`
25  `powers` : ` powers`
26  ` dep` : ` dep`
27  ` on` : ` on`
28  ` our` : ` our`
29  `tutio` : `stitution`
30  `tutio` : `nstitution`
31  `ei` : `ei`
32  ` states` : ` states`
33  `ore` : `ore`
34  `dicia` : ` judicia`
35  `e leg` : `e leg`
36  `onstitu` : ` constitut`
37  `onstitu` : ` constitu`
38  ` consti` : ` constituent`
39  ` w` : ` w`
40  ` to` : ` to`
41  ` to` : ` to `
42  `ity` : `ity`
43  `s, a` : `s, a`
44  `he stat` : ` the state`
45  `he stat` : ` the stat`
46  `the sta` : ` the sta`
47  `and c` : ` and c`
48  `wers ` : ` powers `
49  ` the st` : ` the st`
50  ` a ` : ` a `
51  `the le` : ` the le`
52  `exe` : `exe`
53  `diciar` : ` judiciary`
54  ` di` : ` di`
55  `e state` : `e state`
56  `upon t` : ` upon t`
```

```
57  `the con` : ` the con`
58  ` the l` : ` the l`
59  ` on the` : ` on the`
60  `by the` : ` by the`
61  `ationa` : `ationa`
62  `ationa` : ` national `
63  `thre` : ` thre`
64  `thre` : `thre`
65  ` and co` : ` and co`
66  ` by th` : ` by th`
67  `ative ` : `ative `
68  `he na` : ` the na`
69  `he na` : ` the nat`
70  `re is` : `re is`
71  `re is` : `here is `
72  ` and w` : ` and w`
73  `utive` : ` executive`
74  `s and ` : `s and `
75  `li` : `li`
76  ` more ` : ` more `
77  `eral` : `eral`
78  `eral` : `eral `
79  `tere` : `tere`
80  `icl` : ` article`
81  `e nat` : `e nat`
82  `il` : `il`
83  ` cau` : ` cau`
84  ` cau` : ` cause`
85  `he e` : ` the e`
86  `t ` : `t `
87  ` the co` : ` the co`
88  `us` : `us`
89  `very ` : `very `
90  `e power` : `e power`
91  ` or ` : ` or `
92  `ederati` : ` confederation`
93  `interes` : ` interest`
94  `ed by ` : `ed by `
95  `ause` : `ause`
96  `ause` : `cause`
97  `ventio` : `vention`
98  `ventio` : ` convention`
99  `y the ` : `y the `
```

```
100  `eric` : `merica`
101  `eric` : ` america`
102  ` of t` : ` of t`
103  ` of t` : ` of th`
104  `milit` : ` milit`
105  `the fe` : ` the fe`
106  `ral ` : `ral `
107  `cong` : ` congress`
108  `ly` : `ly`
109  `judi` : `judic`
110  ` ex` : ` ex`
111  `e t` : `e t`
112  `oin` : `oin`
113  `every` : ` every`
114  `e that` : `e that `
115  `e that` : `e that`
116  `ey ` : `ey `
117  `ations` : `ations`
118  ` we ` : ` we `
119  `pe` : `pe`
120  ` would` : ` would `
121  ` would` : ` would`
122  `hen` : `hen`
123  `e ex` : `e ex`
124  `ly ` : `ly `
125  ` execut` : ` execut`
126  `of the` : ` of the`
127  `he a` : ` the a`
128  `it ` : `it `
129  `it ` : ` it `
130  `us ` : `us `
131  ` the ex` : ` the ex`
132  `l pow` : `l power`
133  `oo` : `oo`
134  `r fo` : `r fo`
135  `r fo` : `r for`
136  `t c` : `t c`
137  `hey ` : ` they `
138  `heir` : ` their `
139  `heir` : ` their`
140  ` conv` : ` conv`
144 non-blank lines; 141 entries.
```

```
1  C:\TEST\FEDS.HAM
153342 bytes.
2  C:\TEST\FEDS.JAY
36373 bytes.
3  C:\TEST\FEDS.MAD
151019 bytes.
340734 bytes.
proportion in class 1 = 0.450033677
proportion in class 2 = 0.106748803
proportion in class 3 = 0.443216052
Grams kept = 138

             Substring  Chi-score     Frequencies

1              ` the `  120.221377    2340.  341.  2616.
2                `nd `  119.378378    736.  345.  803.
3               `and `  112.964111    644.  310.  724.
4              ` and `  112.439721    618.  302.  716.
5            ` nation`  90.0909537    97.  59.  44.
6               `our `  88.4346452    70.  43.  19.
7     ` confederacies`  87.9571833    5.  18.  4.
8             ` upon `  82.688867    77.  1.  2.
9        ` department`  79.1442903    7.  1.  83.
10  ` national govern`  74.6976514    16.  21.  4.
11              `tive`  74.0448425    103.  9.  226.
12            `racies`  73.233958    5.  18.  8.
13           ` depart`  70.8446694    10.  2.  84.
14             ` stat`  66.8863333    176.  28.  328.
15            ` state`  66.2388805    174.  28.  325.
16      ` legislative`  66.2299353    10.  0.  76.
17            ` power`  65.2177448    106.  13.  226.
18       ` confederac`  62.1917692    16.  23.  13.
19         ` america `  60.1216997    8.  19.  12.
20              `rs o`  58.4625465    27.  6.  108.
21         ` legislat`  58.4012384    52.  0.  129.
22           ` powers`  57.0080181    25.  5.  102.
23            `rs of `  56.45585    25.  6.  103.
24              ` dep`  55.9566474    29.  7.  110.
25               ` on`  53.8746187    182.  90.  328.
26        `nstitution`  53.5688005    72.  0.  145.
27              ` our`  53.4548563    64.  32.  20.
28         `stitution`  52.8073004    73.  0.  145.
29                `ei`  52.3235677    272.  126.  253.
```

30	` states`	51.8645117	87.	15.	191.
31	`ore`	49.4807911	111.	73.	155.
32	` constitut`	49.2648627	78.	0.	145.
33	` judicia`	49.0452066	3.	1.	50.

Given this project's objective of producing comprehensible rules for text classification, it is hoped that readers will agree that expansions such as `artment' to ` department', `dicia' to ` judicia', `he stat' to ` the state' and `heir' to ` their' represent gains in clarity. (Whether this process leads to a gain or loss in terms of predictive accuracy is the subject of the next section.)

Neither STRETCH nor TEFF, however, eliminate short and apparently non-meaningful substrings altogether, as a glance at the above listing will confirm. Sometimes an extension, or lack of extension, appears curious. For example, looking at number 100 on the list in Table 7.1, it is tempting to suspect an error either in the program of the data: why was `eric' expanded to ` america' in Jay's sample (as would be expected) but only to `merica' in Hamilton's? Several such apparent oddities have been investigated and there usually turns out to be a good reason for them. The explanation for `merica' is that both Hamilton (3 times) and Madison (4 times) used the word `chimerical' in the training sets seen by this program. Thus `merica' is sometimes preceded by `a' and sometimes by `i' in Hamilton's and Madison's work.

A similar case is the retention of `earn' among the MAGS markers as well as ` learn'. This brought to light an uncorrected scanner error (`iearner' in place of `learner') in the text from the journal *Machine Learning*, but in any case `earn' would have been retained as the training text also contained the proper name `Kearns', which ensured that the substring `earn' was not in fact always preceded by the letter `l' in this file.

Clearly this shows that the method is somewhat sensitive to misspellings, typographical errors and the presence of rare words or proper names. As it is desirable that a text classifier should be able to cope with at least some moderate level of spelling &/or typing mistakes, some consideration was given to the idea of modifying the system so that strings were extended if a high enough proportion (85% or 90%, say) of their preceding or succeeding characters were identical. Such a program, however, would inevitably be more complex than STRETCH or TEFF and would require a good deal of fine tuning, so on further reflection it was felt that the effort needed to design and debug a foolproof noise-tolerant text-extender would be better spent on other, more central, aspects of the project[26].

[26] There was also a suspicion, which admittedly cannot be rigorously justified, that such a diversion of effort might even work against the serendipitous discovery of marker substrings that lack face validity but nevertheless work well in practice.

So it was decided to settle for the simplicity of STRETCH and TEFF, both of which do, after all, offer an improvement in intelligibility over the basic Monte-Carlo feature-finder (CHISUBS).

7.2 Performance of Extended Text Fragments

The next step was to test whether this improvement had been bought at the cost of reducing the accuracy of classification.

For this purpose the robust Bayesian classifier (MAWS) of section 6.1 was re-run on the 13 benchmark problems using five different sets of marker substrings.

Table 7.2 -- Marker Sets and their Derivation.

Marker-set Label	Derivation
SUB1	13 sets of substrings produced by CHISUBS (exactly the same as described in subsection 6.1.3)
STRETCH1	extended substrings produced by running STRETCH on above markers (SUB1)
SUB2	fresh sets of substring markers produced by CHISUBS, just to test whether random variability is important
STRETCH2	Extended markers produced by running STRETCH on the SUB2 markers
TEFF	extended substring sets produced by TEFF (without using CHISUBS)

Thus the experimental design had 5 conditions (5 sources of marker substrings) used on 13 problems, with 5 replications to smooth out random variation. The four response variables measured were:

Varsused	number of substrings kept by MAW1 after its genetic search for an efficient subset of markers;
Percent	success rate of MAW2 on unseen test data;
Log2pen	logarithmic penalty measure to the base 2 (entropy score) computed by MAW2 on unseen test data;
Precall	precision/recall score computed on unseen test data.

As in previous chapters the median value from five runs was taken as representative for each combination of problem and condition.

The figures quoted in Table 7.3, below, to summarize the results of this trial are means of medians (median of five runs on each problem, averaged over 13 problems).

Table 7.3 -- Summary of Results for 5 Different Marker Sources.

Condition	Varsused	Percent	Log2pen	Precall
SUB1	26.54	74.10	-1.20	.7166
STRETCH1	25.38	73.76	-1.25	.7210
SUB2	26.23	73.12	-1.24	.7135
STRETCH2	25.08	71.39	-1.40	.7026
TEFF	27.23	73.00	-1.36	.7192

Table 7.4 -- Analysis of Variance Summary.

Response Measure	F-ratio (d.f. = 4,60)	Associated Probability
Varsused	1.98	0.109
Percent	1.51	0.210
Log2pen	2.60	0.045
Precall	0.70	0.592

As usual, problem number had a hugely significant effect on all these variables if treated as a factor, so deviations from the mean of each problem were first calculated and then a 1-way analysis of variance with 5 levels performed for each variable.

These results indicate no significant main effect of marker source on number of markers used, percentage success rate or Precall score; but a marginally significant effect on logarithmic penalty score.

This latter was investigated further by computing Tukey's multiple comparison threshold (Hayter, 1984) for the 10 possible pairwise comparisons of Log2pen. Setting an overall alpha value of 0.05 (5% significance level) yielded an alpha for individual comparisons of 0.00662 and at this adjusted level **none** of the 10 comparisons between conditions was significant.

However, using Dunnett's method (Bechhofer & Dunnett, 1988) to compare each of the other four conditions with condition 1 (i.e. using the SUB1 condition as a reference point) gave one significant result at the 5% level, namely that condition 4 (STRETCH2) was worse than condition 1 (SUB1) on this measure. A 95% confidence interval for the difference between these two conditions on Log2pen spans from -0.3869 to -0.0147.

To sum up, as far as Varsused, Percent and Precall are concerned there are no significant differences between conditions. In particular, one collection of randomly generated CHISUBS marker lists may be expected to give broadly similar results to another set (SUB1 versus SUB2), which is rather crucial. Moreover, there is no clear difference between using the raw substrings from CHISUBS and the extended versions of them derived using STRETCH (SUB1 versus STRETCH1, SUB2 versus STRETCH2). Finally, marker strings produced and `stretched' by TEFF in a single program appear to perform similarly to those obtained by a 2-step process involving CHISUBS followed by STRETCH.

As far as Logarithmic Penalty is concerned, the results are borderline. There is some evidence that the STRETCH2 substrings performed worse than the other sets on this measure (mainly attributable, in fact, to very poor results on the LIT2 and SNOB problems). Whether a null-hypothesis test is strictly appropriate here is arguable, and whether adjustment ought to be made to the 5% significance level to compensate for effectively performing four tests with much the same intent side by side is also arguable; but, leaving aside statistical pedantry, it seems fair to conclude that there is some doubt over the reliable equivalence of extended substrings and the substrings they were derived from -- if logarithmic penalty is the main criterion of performance.

Broadly speaking, there is no evidence of a **gain** in performance from using extended substrings, but there is little hard evidence of a loss of performance either. Because TEFF substrings are less open to the criticism of unnatural segmentation that those from CHISUBS, and because they appear on four important measures to give very similar performance to unextended CHISUBS strings, and also because there is a slight question mark over one of the marker sets produced by STRETCH, it was decided to proceed using TEFF as the primary source of marker substrings from here on, including the experiments reported in the next chapter.

Chapter 8

AN EVOLUTIONARY APPROACH TO TEXT CLASSIFICATION

> "Natural selection has superb tactics, but no strategy -- but tactics, if pursued without thought for long enough, can get to places which no strategist would dream of." -- Steven Jones (1994).

After the interlude of Chapter 7, this chapter evaluates an attempt to build on the work of Chapter 6 by combining the best elements of both systems described there -- using evolutionary search along with Bayesian reasoning as in MAW1 while generating intelligible descriptions as in TEXTREE. It thus describes the final attempt in this project to create an effective generic text classification system that produces intelligible stylistic descriptions.

Of all the systems developed as part of this project, the system described here pays most serious attention to devising a language suitable for **representing** stylistic habits.

8.1 TOGA : A Text-Oriented Genetic Algorithm

TOGA (Text-Oriented Genetic Algorithm) is part of a suite of programs called GLADRAGS (Genetic Learning Algorithms Discovering Relational Algebraic Grouping Signatures) devised in an attempt to knit together lessons learned in earlier phases of this project with experience gained during development of the BEAGLE rule-learning package (Forsyth, 1981; Forsyth & Rada, 1986), and thereby produce an effective text-oriented inductive classifier.

Specifically, a twofold goal was that GLADRAGS should be at least as effective (operationalized in terms of performance on the 13 benchmark problems) as MAWS, the robust Bayesian classifier of Chapter 6, while producing decision rules, composed of symbolic relational expressions, at least as comprehensible as the discrimination trees generated by TEXTREE (section 6.2).

8.1.1 System Outline

The system consists of three main modules, written in C, which are executed in sequence.

TOGA (Text-Oriented Genetic Algorithm): this takes marker substrings such as those found by TEFF (section 7.1) and uses an evolutionary search strategy to find relational expressions, involving textual markers, that discriminate among the different types of training texts.

CAPE (Calibrator And Probability Estimator): this combines individual rules produced by TOGA into what may be termed a rule base, collates statistics for later probability estimation, and at the same time filters out rules that contribute little to the overall performance of the rule base.

COAT (Classifier Of ASCII Texts): this applies probabilistic rules that have been combined and calibrated by CAPE to classify seen or unseen text samples.

The TOGA program uses the same genetic search algorithm as MAW1 and IOGA, described in section 5.2. (See also Appendix C.) This is applied to rule structures encoded as fixed-length byte strings. Its operation will be discussed further after the representation scheme used has been explained in 8.1.2.

The CAPE program performs a function that is in some ways analogous to pruning a discrimination tree: TOGA deliberately generates slightly more rules than are expected to be wanted, and CAPE has the task of combining those that work well together and dropping the rest. In addition it collects information that COAT will need to make probability estimates.

Whereas TOGA and CAPE use only training data files, COAT is typically applied to the test sets. It applies the combined rule base produced by CAPE to text samples to classify them. It also collects performance statistics, on the basis that the correct categorization of each text sample is actually known.

8.1.2 Rule Representation Scheme

In a genetic or evolutionary approach to machine learning, the formalism chosen to represent potential solutions must achieve a compromise between two conflicting demands. On the one hand, as emphasized above, the formalism should be comprehensible to human users; on the other hand, it must be amenable to the genetic operations, primarily crossover and mutation, which chop up and recombine potential solutions. These work best with fixed-length strings using an alphabet with low cardinality. Indeed Holland (1975) and his followers (e.g. Holland et al., 1986;

Goldberg, 1989) have concentrated almost exclusively on representations based on fixed-length strings using a binary or ternary alphabet.

For this project a novel rule-representation scheme was devised that attempts to balance these conflicting demands.

Seen from the outside, a TOGA rule is a relational expression composed of four types of element: constants (in this case, small integers), variables (i.e. marker substrings), monadic operators (such as Tanh, the hyperbolic tangent function) and dyadic operators (such as plus and minus). A simple example, derived from the MAGS training data, follows.

```
( `learn` \  48) > (Root ` language` -  `oblem `)
```

This is an algebraic inequality involving three marker substrings, `learn', ` language' and `oblem ', a constant, 48, two dyadic operators, \ and -, and a monadic or unary operator, Root.

All TOGA rules are in fact inequalities, as above, using the relation greater-than as the highest level operator. This arrangement was intended to reduce epistasis -- the idea being, roughly speaking, that components favourable to positive instances would congregate on one side of the `>' sign with items favourable to negative instances congregating on the other side.

In evaluating such expressions, constants represent themselves, marker substrings are replaced by the number of occurrences, in the current line, of the substring concerned, and the arithmetic operators are as in Table 8.1. Note that although variables can here have only integral (counted) values, all calculations are performed internally using double-precision floating-point arithmetic.

Table 8.1 -- Dyadic Operators Used by GLADRAGS.

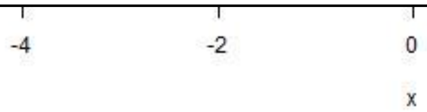
Dyadic Operator	Meaning
>	logical greater than, always used as top-level relational operator
+	addition
-	subtraction
*	multiplication
\	minimum: lesser of left or right operand
^	maximum: left or right operand, whichever is greater

It will be noted that arithmetic division is missing from this list. Various ways of dealing with the possibility of division by zero have been proposed, in connection with evolutionary optimization of arithmetic/logical rules. Schoeneburg's system (1994) traps this event and penalizes expressions that cause it so that their fitness is low and they usually fail to propagate. Koza (1992) recommends using `protected division' which is defined as division unless the denominator is zero, in which case the resultant value is 1. There is also the option of setting the result of a division by zero to a very large number (`pseudo-infinity') and continuing as if nothing exceptional has happened. Preliminary trials of both the latter two expedients suggested that they were more trouble than they were worth. Redefining the division operator tends to introduce counter-intuitive discontinuities which are exploited by the genetic search mechanism but give rise to rules that are hard to fathom, thus undermining the main benefit of using explicit symbolic rules. Accordingly, the division operator was simply omitted[27].

Monadic take precedence over dyadic operators. A list of monadic operators appears as Table 8.2.

[27] My personal view on this matter is that, when creating a rule-representation language, the trick is knowing what to leave out. Austerity in this respect is usually a good policy.

Table 8.2 -- Monadic Operators Used by GLADRAGS.

Monadic Operator	Meaning
[blank]	 (leave operand alone)
Fabs	(floating-point) absolute value of operand, i.e. ignoring minus sign if present
Root	\`safe' square root operator, computed as sqrt(x) if x is zero or positive but as -sqrt(-x) if x is negative
Slog	\`safe' natural logarithm defined as ln(fabs(x) + 1) if x is non-negative, -ln(fabs(x) + 1) otherwise
Tanh	hyperbolic tangent function

The last three monadic operators in Table 8.2 are all, in effect, \`squashing functions': they take an unrestricted numeric value and reduce it to a value within a narrower range which is still monotonically related to the original value. Tanh, for instance, sometimes employed in this fashion as an alternative transfer function to the sigmoid in neural nets (Wasserman, 1993). A graph of this function appears in Figure 8.1.

A variety of numerical \`squashing functions' were thought to be useful in a rule language designed with text classification in mind principally because frequencies of occurrences of words or short substrings in chunks of text of a given size tend to have positively skewed distributions. Since counts are never negative, applying any of the above functions to a variable based on counting substrings has the effect of reducing positive skew. (As a side effect, when applied to whole numbers, they also allow the system to find a range on non-integer constants.)

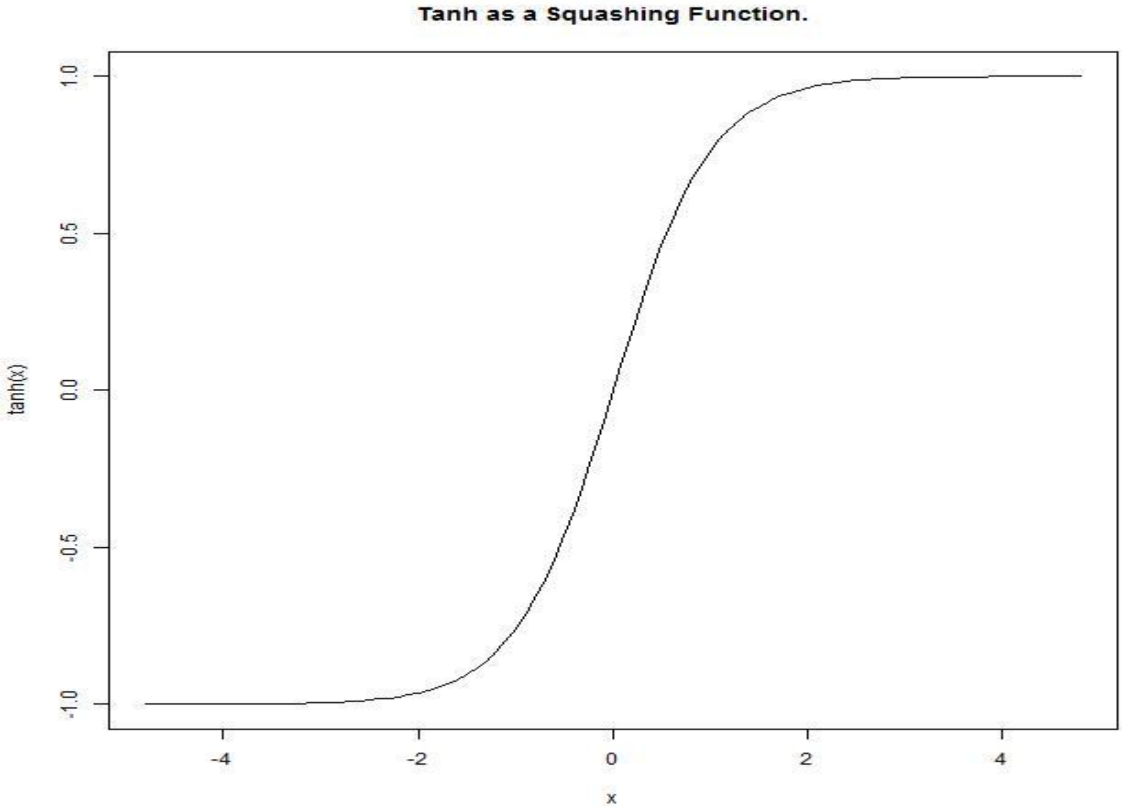


Figure 8.1 -- Graph of Tanh.

Returning to the example expression given earlier,

```
( `learn`  \  48) > (Root ` language` - `oblem `)
```

it is now possible to explain that, when applied to a given line of text, it will yield a true result only if the lesser of the number of occurrences of `learn' and 48 exceeds the the result of subtracting the number of occurrences of `oblem ' from the square root of the number of occurrences of ` language'. When true, this rule indicates that the line concerned very probably comes from the journal *Machine Learning* rather than *Literary & Linguistic Computing*.

Internally, rules of the type described above are represented as binary trees. In fact, they are represented using a data structure known as a heap from its rôle in the Heapsort procedure (see, for instance, Sedgewick, 1988). A heap is a full balanced binary tree held in a vector. The advantage of this form of representation is that it obviates the need for pointers. If heap[j] is a node at a particular position in such a data structure then heap[2*j] is the left subtree of node j and heap[2*j+1] is its right subtree.

A minor complication is introduced in the current application because rule elements go together in pairs, each operand with its associated monadic operator, but this essentially only requires the second indexing formula above to become [j*2+2].

Using this data structure permits a simple mapping from symbolic expressions at the user level to fixed-length gene-strings at the level of the genetic search. The rule evaluation procedures traverse a tree which is treated as having expressions with possibly nested subexpressions, i.e. a hierarchical organization. The genetic search procedures, meanwhile, manipulate a `flat' one-dimensional byte-string. The important point is that the genetic operators do not have to be modified to take cognizance of the syntax of the rule language.

As illustration, the internal format of the rule quoted above is shown in Table 8.3.

Table 8.3 -- Example Rule in Internal Format.

Location	Contents	Interpreted Form
0	146	
1	171	>
2	123	
3	149	\
4	71	
5	71	-
6	115	
7	2	\`learn\`
8	21	
9	151	48
10	100	Root
11	91	\` language\`
12	55	
13	90	\`oblem \`
14	89	
15	87	

In this table, the column labelled `Location' gives the position in the genestring (which in this case has a length of 16 bytes[28]), while the column labelled `Contents' gives the numerical value of the byte found at that position. As far as the genetic operators are

[28] In fact, the last two positions in such a structure are always redundant; so here the contents of locations 14 and 15 have no effect of the meaning of the rule, though they may enter into the genetic manipulations. (A rough analogy with introns or `junk DNA' suggests itself.)

concerned, these byte values will be copied, combined or mutated as part of a fixed-length string with an alphabet size of 255. But the column labelled `Interpreted Form' shows, in effect, how the rule evaluation and printing routines would interpret the same contents. The hierarchical structure is implicit: the left operand of an operator at location P will be found at location P*2 while the right operand will be found at location P*2+2.

Nodes in the second half of such a tree have subtrees that would be beyond the last element in the heap, so they are used not for dyadic operators (which need descendants) but for constants and variables. This is marked in Table 8.3 by a dashed line which demarcates the upper part of the tree, in which each pair of elements has the form

Monadic-Operator   Dyadic-Operator

from the lower part, where each element has the form

Monadic-Operator   Operand.

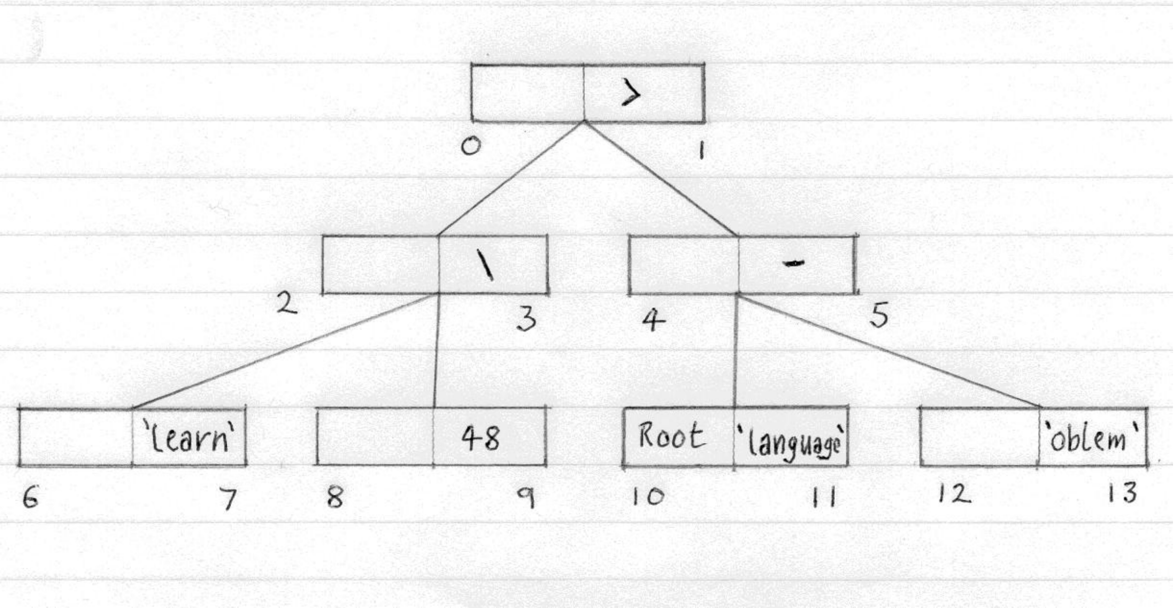


Figure 8.2 -- Heap Represented as a Rule-Tree.

A heap-tree like this can be evaluated recursively, or iteratively in a single backward sweep. Figure 8.2 shows this same expression in a more graphic tree-like representation. It will be seen that rules such as described deliver a Boolean value, true or false. No fuzziness is present as this was thought to mitigate against ready understandability. C code for the rule-evaluation routines, using this heap-based representation, is given in Appendix C.

8.1.3 The Rule Generation Process

The task of the TOGA program (Text-Oriented Genetic Algorithm) is to generate rules of this kind which exhibit a high degree of association between their truth status and the correct category membership of instances given as training data. TOGA actually allows a user to select one of seven fitness functions, i.e. different ways of measuring the strength of such associations. The merits of various fitness functions will be the subject of section 8.3 but, without prejudicing the findings of that section, the discussion here will assume the use of the simplest fitness function, percentage of categorizations that are correct, to illustrate the principles involved.

TOGA uses essentially the same variant of the Genetic Algorithm described in chapters 5 and 6 to search for rules with high success rate. For the runs reported in this chapter, population size was set to 49 and the number of trials was kept at 3200 (slightly more than the 2048 used in chapter 6 or the 1024 used in chapter 5, as TOGA runs faster than MAW1 and much faster than IOGA). It should be recalled that this is an incremental rather than generational genetic algorithm, so 3200 is the total number of genestrings created and tested, not the number of generations. (In a generational scheme with a population size of 49 in which every member of the population was replaced on each cycle, this would correspond to approximately 65 generations.)

As previously, following the lead of Koza (1992), no strenuous effort was made to fine-tune the parameters of the genetic algorithm[29].

The main cycle of TOGA can be outlined by the following pseudocode.

[29] In case this lack of attention to optimizing the parameters of the genetic algorithm itself is misconstrued, it should be repeated that the genetic search procedure used here was crafted especially for this project and is emphatically not just a standard Holland-style GA such as may be picked up on the Internet. Such simplistic software has been criticized by, among others, Keane (1995): "simple public domain GA's are not so robust". The point is that, as mentioned in previous chapters, what is wanted is an adaptation strategy that is not over-sensitive to precise settings of numerous tuning parameters.

```
FOR  each category in the training data
        generate rules for even-numbered cases
        save best rule found
        generate rules for odd-numbered cases
        save best rule found
        ENDFOR
```

Thus TOGA always produces 2*k rules, where k is the number of (mutually exclusive) categories in the training data. Each rule is optimized to discriminate between a particular category and all other categories lumped together.

The split into odd and even training subsets was done to make almost sure that the two rules produced for each class would not be identical, informal trials having indicated that, given a choice between two runs on half the data and a single run over all the data, the former was likely to be a better use of (approximately) the same amount of computing resources. If a biological analogy is wanted as justification this can be seen as a loose analogue of niche specialization, or even evolutionary isolation brought about by continental drift. Eventually CAPE will bring these separated subpopulations into contact, as -- to pursue the analogy -- the collision of North and South America circa 3 million B.C. mixed previously isolated fauna and flora.

8.1.4 Rule Combination and Calibration

As already stated, TOGA produces 2*k rules, where k is the number of text categories. Each of these rules can be seen as a compound condition or meta-feature, based on logical and arithmetic relations between base-level features.

The task of CAPE (Calibrator And Probability Estimator) is to bring such isolated rules or meta-features together into a single structure, here termed a rule base, that can later be used for probabilistic classification.

Finding the best combination of rules or meta-features is itself an optimization problem, so arguably a genetic algorithm could be used to solve it. However, since the search space by this time is typically quite small, this was not done. Instead, CAPE uses an iterative `greedy' search procedure like that employed in stepwise multiple regression, as outlined in the following pseudocode.

```
Create initial null rule base (just uses baserate probabilities)
FOR  j = 1 to k+1
    R = Propose-Rule
    IF  rule base better with R than before
    THEN  Append R to rule base
    ELSE  Discard R and quit loop early
    ENDIF
    ENDFOR
```

The Propose-Rule subroutine referred to above finds the best as-yet unused rule to try adding to the rule base: it can be outlined as follows.

```
q = very large negative number
FOR  r = 1 to k*2
    IF  rule r is not yet in rule base
    THEN  s = score of rule base with r added
        IF  s > q
        THEN  save s as q; save r as R
        ENDIF
    ENDIF
    ENDFOR
Return R
```

Two main issues need to be resolved before pseudocode such as above can be implemented in a working program:

(1) what is a good way to measure the quality of a rule base (i.e. of a combination of rules)?
(2) within what sort of inferential framework should rules be combined?

Five different measures of rule-base quality (point 1) were tested. The simplest of these, again, is percentage correct categorizations. The more complex ones will be described in section 8.3.

As for types of inference mode (point 2), three different methods were implemented and tested. Section 8.4 describes empirical comparisons between them. They all have in common the use of Bayes's theorem to compute posterior probabilities from baserate probabilities together with conditional probabilities derived from data frequencies.

These three inference modes are referred to from here on as:

(1) Naive Linkage;
(2) Sequential Linkage;
(3) Tabular Linkage.

The adjective naive in this connection seems fairly well established, see, for example, Michie et al. (1994). It is not strongly pejorative.

The Naive-Linkage mode uses the simplifying assumption

$$P(\_R_j \mid C_k) = \prod P(R_j \mid C_k)$$

where R_j signifies the value of the jth rule (0 or 1) and C_k denotes a particular class or category. (The underscore in the equation above should be the logical conjunction symbol, like an inverted 'V', Unicode 2227 hex, 8743 decimal; but I'm damned if I can find a way to persuade MS/Word to put it back in after it gratuitously changed it on conversion from WordPerfect.)

In other words the conditional probability of the logical product of the observed rule values is taken as the numerical product of their individual conditional probabilities. This simplification is commonplace in practical Bayesian reasoning systems (Naylor, 1989; Gammerman & Thatcher, 1991; Langley, 1993) -- including MAW1 and MAW2 described in chapter 6. However, only if the rules are conditionally independent (a most implausible supposition in such a context) is it strictly correct.

Modes 2 and 3 (Sequential and Tabular Linkage) allow for dependencies among rules in different ways. Mode 2 (Sequential Linkage) links the rules using the decision-list formalism introduced by Rivest (1987). In a decision list the order of rules matters. Essentially it is a linear IF-THEN-ELSE structure: a case is evaluated by dropping down the rule list until a true rule is encountered. When a true rule is encountered the case is categorized by reference to the class frequencies held at that position in the list, which are obtained from the cases that arrived at that position in the list during training.

A decision list may be seen as a `degenerate' discrimination tree where right branches always have a depth of 1, giving a less `bushy' structure than in a typical discrimination tree. In natural language, people seem to prefer long chained structures ("the goat that kicked the dog that chased the cat that caught the bird that ate the spider that wriggled and tickled and tickled inside her") to deeply nested ones ("the spider that the bird that the cat that the dog that the goat kicked chased caught ate wriggled and tickled and tickled inside her"). So decision lists were used here on the presumption that their lack of nesting would make them more comprehensible than a tree structure.

In Bayesian terms, using a decision list means that training-data frequencies for

$R_1 = 1$
$R_1 = 0$ & $R_2 = 1$
$R_1 = 0$ & $R_2 = 0$ & $R_3 = 1$ etc.

must be accumulated, where R_j refers to the jth rule. From these, conditional probabilities for a given combination of rule values may be calculated. Additionally, this method must have an extra row in the frequency table to cover cases where no rule is true, i.e. where a case falls right through the decision list.

Finally, Mode 3 (Tabular Linkage) gathers frequency information for all possible combinations of retained rules' truth values. This requires a table of size 2^n, where n is the number of rules kept. Clearly it is the most demanding of space. Moreover, the frequency table associated with the rules will tend to be very sparse if more than a handful of rules are kept, implying that probabilities estimated from the tabulated frequencies may well be statistically unreliable. However, it does implement the correct way to do Bayesian inference when rules are not independent. It automatically accommodates whatever interdependencies are present in the training data. Provided that the training sample is representative of the population of interest, it will give the most accurate results.

A small illustrative example is given below in Table 8.4. Here two rules, R and S, have been kept, and there are 2 categories (C=0, C=1). The training data consists of 244 cases, 116 with C=0 and 128 with C=1. We suppose that we are interested in the probability of R=0 and S=0 given C=1 (presumably as a step towards calculating the probability of C=1 given R=0 and S=0).

Table 8.4 -- Simple Illustration of Three Inference Modes.

Naive Linkage

	C=0	C=1
base frequencies:	116	128
R = 1	5	118
S = 1	18	121

```
P(R=0 & S=0 | C=1)
      = 10/128 * 7/128 = 0.00427
```

Sequential Linkage

	C=0	C=1

```
base frequencies:        116   128
R = 1                      5   118         [IF  R
R = 0 & S = 1             16     8         [ELSE IF  S
R = 0 & S = 0             95     2         [ELSE ....

P(R=0 & S=0 | C=1)
      = 2/128 = 0.0156
```

Tabular Linkage

```
                         C=0   C=1
base frequencies:        116   128
R = 0 & S = 0             95     2
R = 0 & S = 1             16     8
R = 1 & S = 0              3     5
R = 1 & S = 1             21    13

P(R=0 & S=0 | C=1)
      = 2/128 = 0.0156
```

In this instance, modes 2 and 3 give the same answer. However, this need not be the case: the essential difference between sequential and tabular linkage is that the former collapses together some of the rows that would appear in a full table. The essential difference between Naive Bayesian inference and the other two modes is that it normally gives the wrong answers! As seen here, its probability estimates tend to be more extreme (nearer to 0 or 1) than they really should be.

It is, however, the simplest method and it uses information from all the rules (all those that have been kept by CAPE, that is) unlike sequential linkage, which may sometimes take account of only one out of several rules in computing a probability estimate, thus ignoring potentially useful information. In addition, as it is commonly used by other researchers (as well as by earlier programs in this series) it provides a useful baseline. So, although modes 2 and 3 were expected to give better results, mode 1 was not excluded from consideration.

To conclude this brief sketch of the operation of CAPE and COAT, it should be added that both programs, in whichever inference mode they are working, add 1 to each cell of the frequency table associated with the rules before computing conditional probabilities. The effect of this is to reduce slightly the extremity of the calculated probabilities. It also prevents division by zero in empty rows.

8.1.5 A Framework for Experimentation

The design of GLADRAGS allows some pertinent issues in this field to be explored.

As with any complex piece of software, there is a plethora of parameters whose settings could be varied. These include parameters of the genetic algorithm such as population size, mutation rate and so forth; but, as explained earlier, these are not salient at present so they were not further studied here. This still leaves plenty aspects of the system whose behaviour might repay further investigation. Among these, it was felt that three in particular led to questions that were worth trying to answer:

- ➢ what sort of fitness function works best?
- ➢ how does rule complexity affect rule-base performance?
- ➢ which inference mode appears to give best results?

Choice of fitness function and representation are always key issues in genetic or evolutionary computing, the latter especially so here in view of the present project's emphasis on representing stylistic habits. Inference mode is also important, since finding a computable method that overcomes known deficiencies in the naive Bayesian reasoning process would be a worthy objective in itself.

Because of the architecture of the GLADRAGS suite the term fitness could refer to two distinct measures of quality:

(1) quality of individual rules (in TOGA);
(2) quality of a combined rule base (in CAPE).

To avoid confusion, measures of the former will be referred to in this chapter as <u>fitness</u> functions, while measures of the latter will be termed <u>quality</u> scores. Although sometimes the same mathematical formula can be used for both purposes (the simplest being success rate) they should be distinguished as it is not logically necessary that the same function will work best for both purposes. (And there are some functions that do not readily cover both requirements.)

The next three sections (8.2 to 8.4) discuss results obtained from employing the GLADRAGS system as a vehicle to investigate these issues empirically (using the 13 benchmark text-classification problems). However, with 7 plausible fitness functions, 5 quality measures, various possible rule lengths (16, 32, 64 bytes), the possibility of switching on or off the use of monadic operators in the rules, as well as three different inference modes to be investigated .... it will be appreciated that a single exhaustive multi-factorial experiment would be impractical[30]. Instead the `search space' (for that

[30] A full or `brute-force' factorial study with five replications on 13 benchmark

is what it is) was partitioned into two partly overlapping areas. Thus fitness and quality measures are the main focus of section 8.3. Then, having arrived at a reasonable choice of fitness function and quality score, issues of representation and reasoning are treated in section 8.4.

Even with this subdivision, which sacrifices the possibility of studying some potential high-order interaction terms in order to make the investigation tractable, these experiments consumed many hours of runtime.

8.2 Preliminary Test on Non-Textual Data

Prior to the main experiments, however, an initial proving run was conducted on two numeric data sets. This was because it was considered wise, before embarking on trials whose results were unpredictable, to conduct a small-scale test on a couple of well studied classification problems.

8.2.1 Data Used

The first of these is one of the best known multivariate data sets in the literature, the Iris data already used in Chapter 5. Although widely known as "Fisher's Iris Data" after two papers by Sir Ronald Fisher (1936, 1938) in which the technique of linear discriminant analysis was developed, these data were, according to Ripley (1993), actually gathered by Anderson (1935). Each record in this data set describes an iris flower, measured on four features: sepal length, sepal width, petal length, petal width. These plants come from three different subspecies: Iris setosa, versicolor and virginica. There are 50 examples of each type.

The second problem chosen was, for variety, one that had not been used in Chapter 5. It is a medical data set originally analyzed by Reaven and Miller (1979) and reproduced in Andrews & Herzberg (1985) and in Lunn & McNeil (1991). It contains clinical measurements on three groups of hospital patients collected as part of a study into the relationship between chemical subclinical diabetes and overt diabetes in 145 non-obese adult subjects. The fields present in each case of the file were:

problems would, under quite realistic assumptions, require testing of: 7 fitness functions, 5 quality scores, 3 rule sizes, and 2 states (presence/absence) of monadic operators on 13 problems, with 5 replications -- each involving the running of three programs. The product of these numbers comes to 40,950, which, even though the programs are quite fast, would be a somewhat daunting prospect.

ID	patient identification number;
wt	relative weight of patient;
plasglu	fasting plasma glucose level;
glucose	glucose area;
insulin	insulin area;
SSPG	steady state plasma glucose;
clincode	clinical classification.

Clincode is the target category. It has three values: 1, overt diabetic; 2, chemical diabetic; 3, normal.

As these data sets both use traditional numeric feature-vector format, whereas GLADRAGS works with lines of text, a short Spitbol program was written to convert from numeric to textual form. This was done simply by reproducing the name of each field a number of times proportional to the numeric value of that field. To be precise, n = round(k × x) 8copies of each field name were concatenated into the textual version of each record, where x is the value of the variable concerned and k a user-supplied constant. With the Iris data, this constant k was 10 for all four variables. Thus, for instance, if the value of sepal width were 2.4 then 24 copies of the string `sepwidth' would be included in the line representing that case -- and likewise with the other three features.

Values of k used with the seven fields in the Diabetes data were:

ID	0
wt	10
plasglu	0.1
glucose	0.01
insulin	0.1
SSPG	0.1
clincode	0.

Unlike the scaling of the Iris data, this scaling did involve some loss of precision, due to rounding.

As a method of classifying data held in numeric feature-vector format, this is certainly rather long-winded; but it does allow a preliminary check on the effectiveness of the GLADRAGS suite. To do this, both data sets were split randomly in approximately 60/40 ratio into training and test files.

Table 8.5 gives the result of using COAT (after running TOGA and CAPE) on the unseen part of the Iris data, using default settings for all main parameters except Inference Mode, which was mode 2 -- Sequential Linkage.

Table 8.5 -- Specimen Output Listing: COAT on Iris Data.

```
COAT started on : Mon Jul 03 20:43:28 1995
Rules read in :

(Fabs 2 ^  `Sepwidth`) > ( `Petleng` -  0)

( `Petleng` -  `Sepleng`) > (Slog 3 -  `Petwidth`)

4 markers actually used :
`Sepleng`
`Sepwidth`
`Petleng`
`Petwidth`

Main frequency table, with 3 rows :

     85     2
     32    27    26
1    32     0     0
01    0     2    26
00.   0    25     0

Summary of test results :
            P(Setosa)  P(Versi)   P(Virgin)
   1  1       0.9061     0.0466     0.0473   0  0
   2  1       0.9061     0.0466     0.0473   0  0
   3  1       0.9061     0.0466     0.0473   0  0
   4  1       0.9061     0.0466     0.0473   0  0
   5  1       0.9061     0.0466     0.0473   0  0
   6  1       0.9061     0.0466     0.0473   0  0
   7  1       0.9061     0.0466     0.0473   0  0
   8  1       0.9061     0.0466     0.0473   0  0
   9  1       0.9061     0.0466     0.0473   0  0
  10  1       0.9061     0.0466     0.0473   0  0
  11  1       0.9061     0.0466     0.0473   0  0
  12  1       0.9061     0.0466     0.0473   0  0
  13  1       0.9061     0.0466     0.0473   0  0
  14  1       0.9061     0.0466     0.0473   0  0
```

15	1	0.9061	0.0466	0.0473	0	0
16	1	0.9061	0.0466	0.0473	0	0
17	1	0.9061	0.0466	0.0473	0	0
18	1	0.9061	0.0466	0.0473	0	0
19	00.	0.0508	0.8940	0.0552	1	1
20	00.	0.0508	0.8940	0.0552	1	1
21	00.	0.0508	0.8940	0.0552	1	1
22	00.	0.0508	0.8940	0.0552	1	1
23	00.	0.0508	0.8940	0.0552	1	1
24	00.	0.0508	0.8940	0.0552	1	1
25	00.	0.0508	0.8940	0.0552	1	1
26	00.	0.0508	0.8940	0.0552	1	1
27	00.	0.0508	0.8940	0.0552	1	1
28	01	0.0456	0.1090	0.8453	2	1
29	00.	0.0508	0.8940	0.0552	1	1
30	00.	0.0508	0.8940	0.0552	1	1
31	00.	0.0508	0.8940	0.0552	1	1
32	00.	0.0508	0.8940	0.0552	1	1
33	01	0.0456	0.1090	0.8453	2	1
34	00.	0.0508	0.8940	0.0552	1	1
35	00.	0.0508	0.8940	0.0552	1	1
36	00.	0.0508	0.8940	0.0552	1	1
37	00.	0.0508	0.8940	0.0552	1	1
38	00.	0.0508	0.8940	0.0552	1	1
39	00.	0.0508	0.8940	0.0552	1	1
40	00.	0.0508	0.8940	0.0552	1	1
41	00.	0.0508	0.8940	0.0552	1	1
42	01	0.0456	0.1090	0.8453	2	2
43	01	0.0456	0.1090	0.8453	2	2
44	01	0.0456	0.1090	0.8453	2	2
45	01	0.0456	0.1090	0.8453	2	2
46	01	0.0456	0.1090	0.8453	2	2
47	01	0.0456	0.1090	0.8453	2	2
48	01	0.0456	0.1090	0.8453	2	2
49	01	0.0456	0.1090	0.8453	2	2
50	01	0.0456	0.1090	0.8453	2	2
51	01	0.0456	0.1090	0.8453	2	2
52	01	0.0456	0.1090	0.8453	2	2
53	01	0.0456	0.1090	0.8453	2	2
54	01	0.0456	0.1090	0.8453	2	2
55	01	0.0456	0.1090	0.8453	2	2
56	01	0.0456	0.1090	0.8453	2	2
57	01	0.0456	0.1090	0.8453	2	2

```
  58  01     0.0456     0.1090     0.8453   2  2
  59  01     0.0456     0.1090     0.8453   2  2
  60  01     0.0456     0.1090     0.8453   2  2
  61  01     0.0456     0.1090     0.8453   2  2
  62  01     0.0456     0.1090     0.8453   2  2
  63  01     0.0456     0.1090     0.8453   2  2
  64  01     0.0456     0.1090     0.8453   2  2
  65  01     0.0456     0.1090     0.8453   2  2

Success Percentage = 96.923077
Cramer's V = 0.960868
Log-penalty = -0.279533
Brier Score = 0.930415

Confusion Matrix :
    0      18      0      0
    1       0     21      0
    2       0      2     24

Precall =     0.9721

Inference mode : Sequential Linkage
No. of rules used = 2

Fall-out table :

      65     2
      18    23    24
1     18     0     0
01     0     2    24
00.    0    21     0
```

In this table the rules generated by TOGA and selected by CAPE appear near the head of the listing. The main frequency table comes before the summary of results on individual test cases. It gives information about the training data. In this case, there were 85 training cases, 32 setosa, 27 versicolor and 26 virginica. Two rules were kept. When the first rule was true (designated 1 in this output), the frequencies of the three categories were 32:0:0; when the first rule was false but the second was true (designated 01), the frequencies were 0:2:26; when neither rule was true and the data `fell off' the end of the decision list (designated 00.), the data frequencies were 0:25:0.

These counts can be compared with precisely the same form of tabulation at the end of the listing, called a `fall-out table'. This gives the frequencies of the corresponding rule-combinations in the test data, which here consisted of 65 instances. The results are remarkably similar.

Between these tables come the individual test-case results. Here probabilities for each of the three categories are given, in order, in the middle three columns. These are posterior probabilities calculated from the relevant rule combination (1, 01 and 00.) using Bayes's Rule together with the frequencies from the training data. The last two columns show predicted class code then actual class code (0=setosa, 1=versicolor, 2=virginica).

By dividing constants by 10 (to counteract the effect of the scaling described above) and making some simple rearragements and simplifications (such as removing the redundant - 0), this rule-set can be expressed more succinctly as:

```
IF  max(0.2,sepwidth) > petleng
THEN  Setosa
ELSE IF  (sepleng - petleng) < (petwidth - 0.1386)
THEN  Virginica
ELSE  Versicolor
```

In fact sepal width is never less than 0.2 so the expression in the first line could be further simplified to (sepwidth > petleng).

It is interesting to compare this to the pruned classification tree given by Ripley (1993) which, as it happens, makes the same number of errors over the entire data set (4 out of 150).

To make the comparison fair, this is shown not as a tree diagram, but translated into the same conditional rule notation as used above, using the same variable names.

```
IF  petleng < 2.45
THEN  Setosa
ELSE IF  petwidth < 1.75
THEN          IF  petleng < 4.95
              THEN  Versicolor
              ELSE  Virginica
ELSE  Virginica
```

Although both rule sets are fairly simple, it is hoped that this highlights the main point of contrast between these two representation schemes. The discrimination tree uses simpler tests at each node: to pay for this, it embeds them within a more complex organizing structure. Whether the balance of advantage lies with one side or the other in this trade-off is a question that will not be conclusively settled by the end of the present chapter; but at least the GLADRAGS representation scheme (which I believe is novel) enables this issue to be brought into the open for critical examination.

8.2.2 Results

For comparison, the 1-NNC algorithm (described in Chapter 5) was used on these two data sets, as well as a linear discriminant function (LDF) developed using the Minitab statistical package. Summary results are shown in Table 8.6.

Table 8.6 -- Results of 1-NNC, GLADRAGS and LDF on 2 Test Problems.

	1-NNC (Euclidean Metric)	GLADRAGS	LDF
Diabetes Data	81.43	95.08	90.34
Iris Data	93.15	95.28	97.33

Figures quoted in this table are percentage success rates on unseen data. In the case of GLADRAGS, which is non-deterministic, the figure given is the median success rate of five runs. For both 1-NNC and GLADRAGS, the data was split randomly into training and test sets in approximately a 60/40 ratio, but in the case of LDF leave-1-out cross-validation was used. This actually gives LDF a slight advantage: it is using over 99% of the full data sets to form its discriminant rule while the other methods are using only about 60%.

These results show that we have a workable system. Though, as stated, this is only a small-scale test, it is not completely trivial. Nearest-neighbour classification and LDF analysis are not merely `straw men'. For instance, Weiss & Kulikowski (1991) found that both 1-NNC and LDF were more successful than CART (a classification-tree program) on the Iris data, while the LDF also outperformed the best multi-layer Perceptron architecture (trained using Back-Propagation) that they could find. (Perhaps, given that the LDF technique was developed on this data, this finding is not so surprising.)

In addition, Michie et al. (1994), in one of the most extensive machine-learning benchmark studies yet published, found 1-NNC to be the second best of 23 different machine-learning and statistical classification methods they tested on 22 `real-world' data sets[31]. They also found that LDF performed better than Back-Propagation as well

[31] Michie et al. (1994) are rather coy about endorsing any of the 23 techniques they

as all seven tree-growing algorithms that they tested. This overall superiority included better performance on a different (and larger) data set dealing with diabetic patients.

So it is not unreasonable to continue on the basis that GLADRAGS, though novel and therefore relatively untried, is competitive in terms of classification accuracy with more widely used learning systems.

8.3 Testing Various Fitness Functions

In any application of genetic or evolutionary computing, the choice of fitness function or objective function is crucial. The system will strive to optimize this measure, so it must be a good index of what is actually desired. The main aim of this section was to find a fitness function for TOGA that would work well.

8.3.1 Some Fitness and Quality Measures

As it was expected that choice of fitness function would interact strongly with choice of quality score (used in CAPE), the latter factor was also investigated, meaning, in practice, that an effective combination of fitness and quality measures was being sought. In addition, it was thought likely that inference mode would interact with choice of fitness and quality scores, so two of the three Bayesian linkage modes discussed in section 8.1 -- Naive and Sequential -- were also tested.

The source of marker substrings was always TEFF, as described in Chapter 7. Other factors that were not varied included rule length (kept at 16 bytes) and presence or absence of monadic operators. For this experiment, settings of these parameters were chosen that appeared reasonable *prima facie*, and held constant. As regards monadic operators, the settings used caused inclusion of `low level' monadics (i.e. those applied to constants or variables) but exclusion of `high level' monadics (i.e. those applied to subexpressions).

tested as best overall -- rightly so in view of the fact that different classifiers tend to suit different sorts of data as well as the difficulty of finding a single satisfactory measure of system performance. The above comments, therefore, are my (shorthand) interpretation of the following findings from their study: (1) 1-NNC was ranked first or second best on their 22 test sets more often than all but one other method (DIPOL92); (2) LDF was ranked in the top five more often than all except two other methods (DIPOL92 and ALLOC80).

(Rule length and presence/absence of monadic operators are among the attributes investigated in section 8.4.)

Seven plausible fitness functions were considered. They are listed in Table 8.7.

Table 8.7 -- Fitness Functions Tested.

Code Number	Fitness Function	Formula
0	Success Rate	(a + d) / N
1	Phi Coefficient (Press et al., 1986)	$\frac{ad-bc}{\sqrt{(a+b)\ (c+d)\ (a+c)\ (b+d)}}$
2	Entropy Score (Quinlan, 1986)	$\sum_{j=1}^{c} \log_2 (n_{kj}/n_{.j})$ where k=1 for true category of current case, otherwise k=0
3	Logodds Ratio (Gowers, 1995)	$\log_2 (\frac{ad}{bc})$
4	Product Purity Index (Forsyth, 1995)	$\frac{\prod n_{kj}}{\prod n_{.j}}$ where k=1 for true category of current case, otherwise k=0
5	Modified Gini Index (Breiman et al., 1984)	$\sum_{j=1}^{c} (\frac{n_{0j}}{n_{.j}})^2 + \sum_{j=1}^{c} (\frac{n_{1j}}{n_{.j}})^2$
6	Yule's Q (Porkess, 1988)	(ad - bc) / (ad + bc)

Each of these measures has at one time or another been proposed either as a measure of association in the statistical analysis of contingency tables or as a splitting criterion in a tree-growing algorithm, except number 4 (Product Purity Index) which is an invention of the present author. The Gini Index is `modified' in that it is ordinarily used as an index of impurity, whereas here it is, in effect, inverted, to be used as an index of purity. Details may be found in the references given, except for Success Rate, whose origin is lost in the mists of time.

For methods 0, 1, 3 and 6, the notation used assumes that a cross-tabulation of rule truth-value with positive and negative instances has been made, as in Table 8.8 below.

Table 8.8 -- Notation for Dichotomized Contingency Tables.

	Positive Instance	Negative Instance
Rule True	a	b
Rule False	c	d

Here a, b, c and d are counts. The grand total a+b+c+d is referred to as N. (With a 2-by-2 table like this, the absolute value of the Phi Coefficient -- fitness function number 1 -- is equivalent to Cramér's V (see Press et al., 1986), which in turn is proportional to Chi-squared.)

With methods 2, 4 and 5, which do not rely on dichotomizing the actual class membership, i.e. which use more than two columns when there are more than two categories, the notation n_{ij} has been used above where i refers to rule status (0 or 1, false or true) and j is the category code number. Here c refers to the number of categories. In addition, the notation $n_{.j}$ has been used as shorthand for the total number of cases in column j (i.e. category j).

Note that for technical reasons (as in 8.1.4), each cell in the frequency table was, in effect, incremented by 1 prior to computing any of these scores.

Five different quality scoring measures were also considered. These are listed in Table 8.9 below

Table 8.9 -- Quality Measures Used in CAPE.

Code Number	**Quality Score**
0	Success Rate
1	Precall
2	mean Logarithmic Penalty (base 2)
3	mean Brier score
4	mean Spherical score

Of these measures the first two use only frequency counts; the last three, however, use the probabilities estimated by the classifier and should, in principle, be inherently more sensitive.

Precall, and Logarithmic Penalty (base 2) have already been defined in section 4.4.

The Brier Score (Brier, 1950) has been used for some time, apparently with considerable success (Lichtenstein et al., 1982), to calibrate the forecasts of U.S. weather forecasters, who are required to attach probabilities to their predictions to indicate their degree of uncertainty. Cooke (1991) gives a generalized version of the Brier Score for multiple categories, which he calls the quadratic scoring rule. For a single probabilistic forecast this is computed as follows

$$B=2p_k-\sum_j p_j^2$$

where p_k is the probability assigned by the forecaster to the category of the true outcome. The basic idea is that forecaster gets a reward proportional to p^2 and inversely proportional to $(1-p)^2$, where p is the subjective probability given to the actual outcome. It is, in fact, a generalization of the mean squared error concept.

The Spherical Scoring Rule is another similar measure used in assessing probabilistic judgements made by human experts. It is said by Cooke (1991) to have some advantages over the logarithmic penalty score as a reward function for probability estimators. It is computed as

$$S = p_k / \text{sqrt}(\sum_j(p_j \times p_j))$$

where again p_k is the estimated probability of the actual outcome category.

Logarithmic penalty, Brier Score and Spherical Scoring are all proper scoring rules. That is to say: a forecaster who knows the true probability distribution over categories cannot in the long run profit by over- or under-stating those probabilities. Thus there is no incentive for forecasters not to state their true beliefs about the situation. (Perhaps surprisingly, not all plausible reward regimes have this desirable property.) Proper scoring rules encourage accuracy and informativeness. Here these scoring rules are being used on automatic classification procedures rather than human experts, but the principle is the same.

Of the 35 possible combinations of these fitness and quality measures, six were selected for further study. Table 8.10 lists them, together with their code numbers, which will be used for brevity in the ensuing discussion.

Table 8.10 -- Fitness & Quality Measures Investigated.

Code Number	Fitness Function	Quality Score
00	Success Rate	Success Rate
11	Phi Coefficient	Precall
22	Entropy Score	Logarithmic Penalty
14	Phi Coefficient	Spherical Scoring
43	Product Purity Index	Brier Score
51	Modified Gini Index	Precall

The first three of these were chosen *a priori*: there were grounds to believe that they might, in a sense, belong together. The last three emerged as the best of 8 other candidate fitness/quality pairings that were compared in a small-scale pilot study using only a subset of the 13 test problems. (Hence most of the 35 possible combinations have still not been tested.)

For most purposes, combinations of fitness with quality measures should be treated jointly, so this experiment may be seen as having a 3-factor design:

6	fitness/quality measures;
2	inference modes;
13	test problems;

with five repetitions in each cell. The test-problem factor is, in essence, a nuisance variable. Its effect was very highly significant (some problems are clearly harder than others) but was removed by replacing response variables by deviations from their mean value for each problem. Thus, in effect, this becomes a 2-factor study.

The main response variables, all measured on unseen data, were:

percent	percentage correct classifications;
log2pen	logarithmic penalty (base 2);
brierval	Brier score;
precall	precision/recall.

Note that these serve a dual purpose -- in TOGA &/or CAPE to assess the quality of rules during training, and in COAT to assess the performance of the classifier on test data. Context should make clear which rôle is being discussed.

Additionally, the effect of the two main factors above on number of rules retained and on number of variables used was also looked at.

8.3.2 Summary of Main Results

The main question being asked is threefold: (1) does choice of fitness/quality measures make a difference? (2) does inference mode matter? and (3) does their interaction have a significant effect?

To boil down a large amount of printout, Table 8.11 shows highly summarized results on the four main dependent variables. Only the associated probability for both main effects and their interaction (derived from F-ratios in a 2-way analysis of variance) is shown; but brief comments are also given underneath in square brackets to explicate the import of those effects that have $p<0.05$.

Table 8.11 -- Summary of Four Analyses of Variance.

Response Variable	Fitness/Quality	Inference Mode	Interaction Effect
Percent	0.021 [22 is worse than the rest]	0.015 [Naive Linkage is better]	0.915
Log2pen	0.097	0.000 [Sequential Linkage is better]	0.986
Brierval	0.016 [22 is worse than rest]	0.220	0.991
Precall	0.006 [51 is worse than rest]	0.031 [Naive Linkage is better]	0.749

No significant interaction effects were revealed by this analysis, which is a relief. It means that the effect of fitness/quality measure and that of inference mode act more or less additively; thus they can be dealt with one at a time.

8.3.3 *Effect of Fitness and Quality Measures*

As far as the combination of fitness and quality measures is concerned, it is clear that this factor can make a substantial difference. The results suggest that this is due to the fact that, while most of the pairings chosen are roughly equivalent, it is possible to make an extremely poor choice. The extract given as Table 8.12, using Percent as the dependent measure, is illustrative.

Table 8.12 -- Effect of Fitness/Quality on Percentage (Residuals).

```
ANALYSIS OF VARIANCE ON %resids
SOURCE      DF        SS        MS         F        p
fitqual      5     602.4     120.5      2.66    0.022
ERROR      774   35113.5      45.4
TOTAL      779   35715.9

                                   INDIVIDUAL 95% CI'S FOR MEAN
                                   BASED ON POOLED STDEV
```

```
LEVEL      N      MEAN     STDEV
                                   ---------+---------+---------+--
   0    130     1.056     6.471                      (-------*-------)
  11    130     0.744     6.294                    (-------*-------)
  14    130     0.011     7.088               (-------*-------)
  22    130    -1.690     6.344    (-------*------)
  43    130     0.167     5.681                (-------*-------)
  51    130    -0.288     8.243             (-------*-------)
                                   ---------+---------+---------+--
POOLED STDEV =    6.735                   -1.5       0.0       1.5

Hsu's MCB (Multiple Comparisons with the Best)

    Family error rate = 0.0500

Critical value = 2.23

Intervals for level mean minus largest of other level means

Level   Lower    Center    Upper
                                     --+---------+---------+---------+--
  0    -1.551     0.312    2.175                        (---------*--------)
 11    -2.175    -0.312    1.551                     (--------*---------)
 14    -2.908    -1.045    0.818                 (---------*--------)
 22    -4.609    -2.746    0.000     (--------*-------------)
 43    -2.752    -0.889    0.974              (---------*--------)
 51    -3.206    -1.343    0.520            (--------*---------)
                                     --+---------+---------+---------+--
                                     -4.0      -2.0       0.0       2.0
```

This shows that the main source of variance is the fact that combination 22 (entropy score with logarithmic penalty) is significantly worse, in terms of percentage correct classifications on unseen data, than the other fitness/quality combinations.

This pattern was repeated with Precall as dependent variable, although, with Log2pen as dependent variable, combination 51 (Gini Index with Precall) proved worse than the rest.

To help pick a fitness/quality combination to carry forward to the next section, Table 8.13 was drawn up. It lists the best and worst performing combination on the four main response variables.

Table 8.13 -- Assessment of Fitness/Quality Measures.

Response Variable	Best Combination	Worst Combination
Percent	00 [Success/Success]	22 [Entropy/Log2pen]
Log2pen	14 [Phi/Spherical]	22
Brierval	51 [Gini/Precall]	22
Precall	11 [Phi/Precall]	51

In general, the simple measures do rather well. It seems clear that 22 is a poor choice. Combination 51 is suspiciously erratic and was put aside for that reason. The choice between 00, 11 and 14 remains undetermined. On grounds of simplicity, it was decided to carry combinations 00 and 11 forward to the next stage.

8.3.4 Effect of Inference Mode

Returning to inference mode, an interesting `quasi-interaction' effect appears in Table 8.11. If Percent correct or Precall is accepted as the most suitable measure of performance then Naive Linkage is preferable to Sequential Linkage. However, if Logarithmic Penalty is used to assess performance then Sequential Linkage is very clearly superior to Naive Linkage. It would seem that this reflects a sharper delineation of an effect that was glimpsed in Chapter 6, in which MAWS, using the equivalent of Naive Bayesian Linkage, consistently had higher success rates on unseen data than the TEXTREE system but was less impressive when logarithmic penalty was used as a criterion.

Logarithmic penalty scoring is very sensitive to a small number of extreme errors and, as shown in 8.1.4, Naive Bayesian Linkage tends to give over-extreme probability estimates. In a nutshell, Naive Bayesian Linkage is right more often than Sequential Linkage, but when it is wrong, it is very wrong. The choice between the two depends on whether the user wants `honest' uninflated probability estimates, in which case Sequential Linkage is to be preferred, or just wants to know which category is most likely, in which case Naive Linkage will do better.

8.3.5 Effects on Number of Variables Used

Both factors, choice of fitness/quality measure and inference mode, also affected the number of rules kept and the number of variables kept by CAPE in the final rule base

-- these two quantities being highly correlated. The table below shows an analysis of these effects on a derived variable called `vars/cat', the number of variables per category. (It is only to be expected that classification problems with more categories will require more variables, in general, than ones with fewer categories: using this ratio compensates for that fact.)

Table 8.14 -- Analysis of Variables Used per Category.

[2-way ANOVA by fit.qual & evalmode]

```
Analysis of Variance for vars/cat
Source              DF      Seq SS     Adj SS    Adj MS      F      P
fit.qual             5    268.940    268.940    53.788  40.68  0.000
evalmode             1     62.381     62.381    62.381  47.18  0.000
fit.qual*evalmode    5      3.947      3.947     0.789   0.60  0.702
Error              768   1015.384   1015.384     1.322
Total              779   1350.652
```

[1-way ANOVA by fit.qual]

```
ANALYSIS OF VARIANCE ON vars/cat
SOURCE     DF          SS          MS         F        p
fit.qual    5      268.94       53.79     38.49    0.000
ERROR     774     1081.71        1.40
TOTAL     779     1350.65
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
 LEVEL      N      MEAN     STDEV
                                     --------+---------+---------+----
     0    130     3.215     1.331            (---*--)
    11    130     3.065     1.281          (--*--)
    14    130     4.032     1.052                          (--*---)
    22    130     4.221     0.990                             (--*---)
    43    130     4.091     1.149                           (--*---)
    51    130     2.686     1.251    (---*--)
                                     --------+---------+---------+----
POOLED STDEV =    1.182                     3.00      3.60      4.20
```

[1-way ANOVA by evalmode]

```
ANALYSIS OF VARIANCE ON vars/cat
SOURCE     DF          SS          MS         F        p
```

```
evalmode    1     62.38     62.38    37.67    0.000
ERROR     778   1288.27      1.66
TOTAL     779   1350.65

                                  INDIVIDUAL 95 PCT CI'S FOR MEAN
                                  BASED ON POOLED STDEV
 LEVEL      N      MEAN     STDEV
                                  -----+---------+---------+---------+-
    1    390     3.834     1.261                          (----*----)
    2    390     3.269     1.312    (----*----)
                                  -----+---------+---------+---------+-
POOLED STDEV =    1.287               3.25      3.50      3.75      4.00
```

The effect of fitness/quality measure on variables per category is suggestive of the existence of an optimum: combination 22 (generally the worst in terms of accuracy on unseen cases) uses the most variables while combination 51 (overall the second worst) uses the fewest. The best combinations (00 and 11) use an intermediate number of variables per category. Probably pairing 22 is overfitting while 51 is using too few variables.

It is perhaps also worth noting that the combinations that give rise to the highest rates of variables used per category (14, 22, 43) are those that employ measures based on probabilities as quality scores while those that lead to the lowest rates are those that employ quality scores based on counting.

The effect of inference mode on variables per category is even more clear-cut: mode 2 (Sequential Linkage) uses significantly fewer variables. It is thus more economical than Naive Linkage while performing nearly as well, and will be retained into the next phase of testing for that reason.

8.3.6 The Most Promising Parameter Settings

Finally, to permit scrutiny of some individual results, Table 8.15 gives results on each of the 13 test problems for the combination of parameter settings which appears to be the best of those so far tested, namely:

fitness function	Phi Coefficient
quality score	Precall
inference mode	Naive Linkage.

Table 8.15 -- GLADRAGS Results on 13 Test Problems (Naive Linkage).

Problem Name	Variables used	Success Rate (%)	Log2pen	Precall
LIT1	16	58.18	-1.63	.5322
LIT2	11	71.19	-1.24	.7085
LIT3	14	57.61	-1.44	.5958
FEDS	13	55.34	-1.49	.3943
SNOB	15	69.72	-1.56	.7492
AGE1	10	60.49	-1.08	.5870
AGE2	4	70.00	-1.03	.6708
AGE3	4	66.67	-1.05	.6250
AGE4	10	48.00	-1.87	.4516
MAGS	7	82.91	-0.93	.8185
WAVE	14	57.14	-1.37	.5827
ART1	4	100.	-0.04	1.0
ART2	6	85.15	-0.56	.8513
Mean:	9.85	67.88	-1.1762	.6590

Once again, these figures are from unseen data. For comparability with presentations in earlier chapters, the figures quoted are medians of five replications. The last line therefore gives a mean of medians.

To save space, only these means of medians are shown in Table 8.16, below, which gives the figures for Sequential Linkage with all other parameters the same as for Table 8.15.

Table 8.16 -- GLADRAGS Performance Figures using Sequential Linkage.

Benchmark Summary:	Variables Used	Success Rate (%)	Log2pen	Precall
Means =	7.08	67.63	-1.1169	.6544

On success rate GLADRAGS does better than TEXTREE or 1-NNC, about as well as IOGA, and less well than the Robust (MAWS) or the Basic Bayesian Text Classifier (BBTC). Thus it cannot be hailed as a breakthrough, though it is fair to point out that it does better than MAWS in terms of logarithmic penalty, while, at the same time, using significantly fewer variables on average -- particulary so with mode 2 (Sequential Linkage), as shown in Table 8.16.

8.4 An Experiment on Rule Complexity and Inference Mode

With suitable combinations of fitness and quality measures selected, this section concentrates primarily on the issues of rule complexity and mode of inference. Essentially it recounts a four-way factorial experiment with structure as outlined in Table 8.17 below. (Shorthand identifiers of these factors, for future reference, are also given in square brackets.)

Table 8.17 -- Factors Investigated.

Factor	Levels
Length of Rules [Rulesize]	2 levels: 16 bytes (short); 32 bytes (long)
Monadic Operators [Monadics]	2 levels: `low-level' monadics present; no monadics
Fitness/Quality Measures [Fitqual]	2 levels: 00 = Success Rate with Success Rate; 11 = Phi-score with Precall (see section 8.3)
Inference Mode [Evalmode]	3 levels: 1 = Naive Linkage; 2 = Sequential Linkage; 3 = Tabular Linkage (see section 8.1)

As usual the experiment used the 13 benchmark problems, with 5 repetitions in each cell. Once again the effect of problem number was factored out by using residuals.

The first two factors listed above are associated with rule complexity, giving, in effect, four grades of complexity: (1) short rules without monadics; (2) short rules with monadics; (3) long rules without monadics; (4) long rules with monadics. Grade (2) in this list is the combination already used in section 8.3.

It was hypothesized that there is an optimum rule length; in other words, that it is possible for rules to be too long or too short. The present experiment was designed to point to the range in which that optimum might be found.

The third factor (fitness/quality score) was included to connect this study with that of the previous section, thus allowing at least some assessment to be made of whether the splitting of this investigation into two main parts for practical reasons might have compromised its findings (e.g. by masking potentially important high-level interaction effects).

The last factor above allows a fuller study of the effect of inference mode than in the previous section.

8.4.1 Factors Affecting Accuracy

To summarize the effect of the factors manipulated on accuracy on unseen test data, this subsection considers each of the four main response variables in turn. In each case, as a digest of results obtained from a 4-way analysis of variance, only figures from significant main effects or interaction effects are shown, although some non-significant effects are mentioned when unexpected. (Effects for which no figure is given may be assumed to have an associated probability of more than 0.05.)

Table 8.18 -- Effect on Percentage Correct Classifications.

Effect	Associated Probability	Comment
Rulesize	0.000	Longer rules do better
Fitqual	0.012	11 (Phi-score with Precall) better than 00 (Success Rate with Success Rate)
Fitqual * Monadics	0.004	(See below)
Rulesize * Fitqual * Monadics	0.003	(See below)

There was no significant main effect of Monadics or Evalmode, the latter's associated probability of 0.44 suggesting that all three inference modes are about equally good on this measure.

The 2- and 3-way interactions involving fitness/quality measures, presence of monadic operators and length of rules can be elucidated verbally: it seems that for short rules it is slightly better to have low-level monadic operators; with long rules, it is better to have low-level monadics in fitness mode 00 but much better not to have them in fitness mode 11. Thus rule complexity does interact to some degree with choice of fitness function -- not, unfortunately, in a very simple manner.

Turning to logarithmic penalty as a response variable, there is no need for a table. There was a highly significant main effect of Evalmode ($F_{2,1536} = 21.64$; $p < 0.0005$) with no other factor or interaction even close to significance. Thus the effect of inference mode swamped all other factors tested in determining performance. Specifically, mode 3 (Tabular Linkage) was clearly best while mode 1 (Naive Linkage) was clearly worst.

Table 8.19 gives the equivalent information relating to Brier score as a dependent variable.

Table 8.19 -- Factors Affecting Brier Scores on Test Data.

Effect	Associated Probability	Comment
Fitqual	0.036	11 better than 00 (Phi-score with Precall better than Success Rate with Success Rate)
Evalmode	0.001	Mode 3 (Tabular Linkage) best; Mode 2 (Sequential Linkage) worst.

With this measure, rule length did not seem to make much difference.

Using Precall as the dependent variable gave almost exactly the same pattern as percentage success rate, so is not duplicated here.

8.4.2 Factors Affecting Variable Usage

Table 8.20 displays the results using vars/cat (number of variables used per category) as the dependent measure, in the same form as in Tables 8.18 and 8.19. The fact that longer rules tend to result in retention of more variables is hardly surprising. Less predictable is the fact that Inference Mode has a strong and consistent effect on the number of variables retained by CAPE in the rule base.

Table 8.20 -- Factors Affecting Number of Variables Used per Category.

Effect	Associated Probability	Comment
Rulesize	0.000	Longer rules use more variables per category
Evalmode	0.000	Sequential Linkage uses fewest, Tabular Linkage uses most variables per category
Rulesize * Evalmode	0.034	Effect of Evalmode (above) more pronounced with long rules than short ones

8.4.3 A Suitable Combination of Parameter Settings

A clear message from these results is that Tabular Linkage (inference mode 3) is best. With hindsight, a plausible explanation for this is that Tabular Linkage (like Sequential Linkage but unlike Naive Linkage) performs the Bayesian computations correctly even if there are interdependencies among the rules; in addition (like Naive Linkage but unlike Sequential) it uses information from all retained rules, not just a subset. In short, it does the calculations precisely and does not throw away information -- which cannot be said about the other two inference modes tested. This may well have implications for other systems that automate Bayesian reasoning.

On complexity, it would seem that short rules without monadic operators are over-simplified. None of the other three gradings of complexity is clearly best.

To arrive at a suggested best combination of settings, the results on Brierval (a compromise between the over-sensitivity of logarithmic penalty and the relative crudity of percentage success rate) were looked at for guidance. The parameter combination which gave the best overall mean Brier score (of 0.5652) on unseen test data was:

Inference Mode	Tabular Linkage
Fitness Function	Phi Coefficient
Quality Score	Precall
Rule Length	32 bytes (`long')
Monadic Operators	Absent;

although the Brier scores with low-level monadics present were very close, suggesting that this last setting is optional. Individual results on the 13 benchmark problems, using this combination of settings, are presented for inspection below in Table 8.21.

Table 8.21 -- Test Results with `Recommended' Parameter Settings.

Problem Name	Variables Used	Success Rate (%)	Log2pen	Precall
LIT1	28	58.18	-1.54	.5477
LIT2	22	72.88	-1.07	.7388
LIT3	22	58.70	-1.45	.5956
FEDS	22	51.37	-1.46	.3639
SNOB	24	70.52	-1.44	.7394
AGE1	11	67.90	-1.01	.6370
AGE2	12	80.00	-0.67	.7576
AGE3	8	73.33	-0.89	.7
AGE4	7	64.00	-1.12	.6052
MAGS	9	82.05	-0.74	.8089
WAVE	22	59.52	-1.27	.6079
ART1	5	98.35	-0.12	.9825
ART2	16	81.19	-0.59	.8167
Mean:	16.00	70.61	-1.0285	.6847

8.4.4 Summary

These results show that, as expected, choice of fitness function and complexity of representation can have an effect on the performance of a trainable classifier like GLADRAGS. Perhaps less predictably, they also show that choice of inference mode can be crucial. When logical rules (i.e. compound meta-features or constructed predicates) are used in Bayesian inference, it does appear that it is worth paying the overhead of allowing for conditional interdependencies by collating frequencies in a signature table -- to use a term introduced by an early pioneer of machine-learning research, A.L. Samuel (1967).

Although greedier of memory than the commonly used naive linkage mode, GLADRAGS shows that such a method is quite feasible on a modern personal computer.

One potential advantage of using signature tables in a probabilistic classifier -- which was not in fact exploited here -- is the possibility of flagging never-before or very seldom encountered combinations of meta-features, i.e. of marking as dubious in some way estimates based on rows in the table with small frequency totals. In a real-world application such diagnostic information could be extremely valuable, even though it does not change the number of correct decisions made by the classifier.

Evaluating GLADRAGS as a text classification system on the basis of its average success rate on unseen data from the 13 benchmark problems, it is better than 1-NNC, IOGA or TEXTREE but less good than MAWS (the robust Bayesian classifier of Chapter 6) or BBTC (the basic Bayesian Text Classifier of Chapter 4). If logarithmic penalty is the criterion, then it also does better than the robust Bayesian classifier; i.e. it is the best of the systems tested in this project other than BBTC.

This is a respectable performance level overall, but it fails to achieve the target set in Chapter 4. And its performance on the *Federalist Papers* problem is undeniably disappointing.

Thus it must be said that the attempt to combine efficacy with intelligibility (a central thrust of this project) has been less than fully successful -- although several interesting findings and some useful software have emerged along the way. BBTC is simple and effective, but its `knowledge base' is large and inscrutable. All methods tried here to attain the same effectiveness with smaller and more transparent knowledge bases, including GLADRAGS, have paid some sort of price in terms of loss of performance.

8.5 Does Stricter Selection of Features Help?

One potential source of weakness in both GLADRAGS and MAWS which has not so far been seriously investigated is their dependence on the marker substrings found by the Monte-Carlo feature-finders (either CHISUBS or TEFF). Before concluding this phase of the project, therefore, it was decided to investigate whether refining this feature-finding process would make a significant difference.

During testing of both MAWS and GLADRAGS it became evident that the Monte-Carlo feature-finding programs quite regularly threw up marker substrings which were obviously contextual (such as ` confederacies' from the FEDS data) or were used only in a narrow segment of the training text (such as " they'll stone you" from Bob

Dylan's data), as well as occasional proper names. To a native speaker such discriminators are obviously brittle: they are unlikely to transfer to other contexts.

Of course in a real application this sort of marker can easily be post-edited by a human analyst, but a goal here was to keep the system generic by avoiding reliance on domain knowledge external to the text. So an attempt was made to devise a more sophisticated quality measure for such marker substrings than the Chi-squared score used thus far, to winnow out such blatantly over-specialized markers -- without relying on human intervention.

An ideal textual marker would have the following properties:

(1) Distinctiveness: it is used at different rates between text types;
(2) Commonness: it is fairly commonly used, at least in one type of text;
(3) Consistency: it is used at a relatively stable rate within text types.

The Chi-squared score so far employed to rank substrings, while adequate in assessing distinctiveness (and, to a lesser extent, commonness) is definitely inadequate in assessing stability. It only considers rate of usage over an entire text sample.

To remedy this, a Spitbol program called POSTMARX was written to take markers from CHISUBS, STRETCH or TEFF (or indeed any other source) and subject them to further screening. After casting around for a suitable quality measure, a dual-component score was eventually settled upon, referred to as Iota (Index Of Textual Association) in what follows. Iota is defined by the equation below.

$$\iota = \omega^2 - V^2$$

The first component of this score (omega squared) is a measure of variance accounted for that has been advocated (Smith, 1982) as a way of expressing the strength of a relationship found in an analysis of variance. It serves the same function with respect to an analysis of variance as R^2 does with respect to a regression analysis. In a 1-way analysis of variance it is monotonically related to the F-ratio of the treatment effect. Treating text type as a factor and number of occurrences in each line as the dependent measure it can be computed as an index of how much variability in occurrence of the substring concerned is explained by differences between text types. For the present purpose it has two desirable characteristics: (1) it must have a value between zero and 1 (unlike the F-ratio); and (2) it is sensitive to variance within groups. Of two strings that have the same Chi-squared score the one whose rates of occurrence are less erratic will in general have a higher omega squared. Computing formulas for this measure may be found in Hays (1969) or Oakes (1982).

Despite point (2) above, omega squared does not sufficiently strongly penalize unbalanced rates of usage within a single text type. This aspect is dealt with by the second component of the above formula (V^2) which functions here as a index of inconsistency.

To compute V^2, the text samples are split in half. Next a Chi-squared value is calculated with the expected frequencies computed on the basis that the number of occurrences should be equal in both halves, for each category. Then this Chi-squared value is converted into Cramér's V using the formula given in Press et al. (1986) and in Siegel & Castellan (1988),

$$V=\sqrt{\frac{\chi^2}{N(L-1)}}$$

where L is the lesser of the number of rows and the number of columns in the contingency table. (In the present case, L=2 so the factor (L-1) effectively disappears.)

Cramér's V is an index of association derivative from Chi-squared: it can range in value from zero to 1, like a correlation coefficient. Finally V is squared (by analogy with the squaring of the product-moment correlation coefficient to arrive at R^2, the coefficient of determination) and then subtracted from omega squared to give an overall score.

While some of the above manipulations might be improper within a `classical' hypothesis-testing paradigm, it should be remembered that here both statistics are being used merely as indices, which both happen to give values lying in a convenient range (0 to 1). To obtain a score for each marker substring, what happens is that a measure of split-half unreliability is subtracted from a measure of variance accounted for by text category -- giving an overall score in the interval -1 to +1.

Clearly substrings used with a high degree of inconsistency across a text sample will tend to have a high value of V, but the reverse is not necessarily true: a string that is only used in a narrow block that happens to straddle the half-way point of a text will have a low score on this measure. However, this sort of undetected imbalance is likely to be a rare event. The practical effect of using Iota as a merit score is perhaps best gauged from a look at Table 8.22, a listing of the best 20 and worst 20 of 144 *Federalist* markers according to this index. These substrings were found by TEFF and post-processed by POSTMARX. The middle-ranking items have been removed to save space.

Table 8.22 -- Federalist Substrings Ranked by POSTMARX.

```
POSTMARX output;  date: 07/12/95 11:15:56
gramsize = 17
1  C:\TEST\FEDS.HAM
153342 bytes.
2  C:\TEST\FEDS.JAY
36373 bytes.
3  C:\TEST\FEDS.MAD
151019 bytes.
340734 bytes.
proportion in class 1 = 0.450034338
proportion in class 2 = 0.10674896
proportion in class 3 = 0.443216703

144 substrings ranked, from feds.tff

                          Iota  Chi-square  F-value    Frequencies
1            ` the `   0.18903  120.2212   66.1065  2340. 341.  2616.
2              `nd `   0.17586  119.378203  60.2294  736.  345.  803.
3            ` and `   0.17129  112.439556  58.2735  618.  302.  716.
4               `nd`   0.15778  111.989805  52.623  1139.  469.  1182
5              `and`   0.14419  96.6162237  48.7294  741.  327.  786.
6           ` upon `   0.11385  82.6887457  49.8837  77.  1.  2.
7               `d `    0.1083  71.3790933  34.1886  1856.  644. 1949.
8             ` the`   0.10631  54.6671535  34.508  2864.  537.  3110.
9              `the` 0.9431E-1  47.1473631  30.4021  3034.  610.  3337
10               `d` 0.9366E-1  59.1202102  29.1958  4001.  1218. 4111
11         ` nation` 0.8879E-1  90.0908215  43.9604  97.  59.  44.
12              `an` 0.8073E-1   50.236752  25.881  1690.  545.  1612.
13              `ve` 0.7676E-1   60.557688  28.262  865.  245.  1203.
14               `n` 0.7644E-1  40.3601482  23.6336  9032.  2437. 8775
15             ` to`   0.76E-1  45.7700576  24.1314  1136.  209.  831.
16              `he` 0.7509E-1  36.6658827  24.0081  3261.  689.  3565
17            ` to ` 0.7448E-1  45.1803329  24.0204  1079.  198.  784.
18       ` upon the` 0.7307E-1  36.6477352  22.366   32.  0.  0.
19          ` power` 0.7237E-1  65.2176491  26.7308  106.  13.  226.
20               `e` 0.7121E-1  33.0856977 22.0352  16336. 3964. 17128

[.... 104 lines deleted to save space ....]

-125       ` congress`  -0.22881  26.5563135  9.2534   8.  7.  46.
-126           ` thre`  -0.24655  34.0352466  12.8187   9.  14.  11.
-127       ` members `  -0.27506   34.224591  12.3427   7.  2.  47.
-128   ` the legislat`  -0.28815   41.211705  12.2847   29.  0.  82.
```

```
-129      `e legislat`  -0.29297  48.1936421  15.4261   39.  0.  102.
-130          `e legi`  -0.29553  49.1036789  15.8128   39.  0.  103.
-131          ` milit`  -0.30414  27.2893878  9.2021   67.  7.  19.
-132             `dep`  -0.33363  46.8229311  13.9467   35.  18.  119.
-133        ` britain`   -0.3568  32.9445927  14.3867   8.  12.  7.
-134            `tive`  -0.37746  74.0447338  15.6405   103.  9.  226.
-135  ` confederation`  -0.38016  29.4200431  14.8467   12.  0.  47.
-136     ` convention`  -0.41098   28.291387  10.0564   7.  10.  45.
-137      ` legislati`  -0.52609  67.5837473  23.682   11.  0.  79.
-138   ` legislative `  -0.56219  51.6692207  17.6564   8.  0.  60.
-139         ` judici`  -0.65205   41.418436  13.41   5.  3.  51.
-140        ` judicia`  -0.69152  49.0451347  15.324   3.  1.  50.
-141         ` depart`  -0.74314  70.8445655  16.6722   10.  2.  84.
-142        ` executi`  -0.80437  25.2707165  6.0398   52.  1.  89.
-143     ` department`  -0.83469  79.1441742  18.0174   7.  1.  83.
-144      ` executive`  -0.85191  33.3487543  7.8093   36.  1.  84.

Substrings kept = 99
```

In this listing the three numbers following the substring itself are the quality score, Iota (as described above), the Chi-squared value (which till now has been the only figure of merit used for this purpose) and the F-ratio. Following that are three counts, showing how often the substring occurred in the text samples from Hamilton, Jay and Madison respectively. It will be seen that the rank order by Chi-squared is not preserved by this scoring formula (nor is that by F-ratio).

The leftmost column gives ranks: a minus sign is prefixed to the rank of items with negative Iota scores.

Although based only on information concerning frequencies within the training text samples, this reordering clearly causes a preponderance of contextual words such as `department', as well as proper names such as `britain', to sink to the bottom, a pattern that is repeated, with variations, with most of the 13 benchmark text sets. Grammatical words and fragments, such as `upon' and `and' (a reliable Jay marker), tend to remain high in the list. Subjectively, the method appears to work quite well.

The question is: does this filtering of markers help either GLADRAGS or MAWS to achieve better results?

To test this, all 13 TEFF marker files were post-processed by POSTMARX. Substrings with an Iota value below zero were discarded. (In addition, the condition for excluding rare substrings was tightened up slightly: previously strings occurring less than 11 times had been excluded as too rare; in POSTMARX, currently, strings that occur less than k+13 times, where k is the number of text categories, are excluded.)

As a result, almost 33% of the TEFF markers were discarded altogether. Then MAWS and GLADRAGS were re-run, using the markers that remained. In the case of GLADRAGS, parameter settings were as recommended in subsection 8.4.4.

The results obtained are summarized in Table 8.23. (The figures quoted, as usual, are means over 13 problems of medians over 5 runs of results on unseen data.)

Table 8.23 -- Results with and without Post-Processing by POSTMARX.

Condition	Variables Used	Success Rate (%)	Log2pen	Precall
GLADRAGS / TEFF	16.00	70.61	-1.03	.6847
GLADRAGS / POSTMARX	15.00	67.55	-1.07	.6528
MAWS / TEFF	27.23	73.00	-1.36	.7198
MAWS / POSTMARX	21.15	72.38	-1.26	.7166

In passing, it may be noted that this table highlights the fact that GLADRAGS consistently uses fewer variables than MAWS; it also highlights the fact that GLADRAGS does better in terms of logarithmic penalty while MAWS does better in terms of percentage success rate.

To assess the effect of filtering out apparently unreliable markers, a paired t-test was performed for each of the four response variables on the results obtained by GLADRAGS (TEFF versus POSTMARX strings) and likewise for MAWS. Of these eight tests, one was significant (t = 4.97; p = 0.0003), namely the test on the number of variables used by MAWS: this difference is unlikely to be a chance effect. It seems that post-filtering does cause MAW1 to keep fewer variables.

None of the other measures, however, are significantly affected. In particular, the differences in GLADRAGS between 70.61% and 67.55% success rate and in MAWS between -1.36 and -1.26 on logarithmic penalty are such as might be expected due to random variability.

It seems fair to conclude that POSTMARX does not improve the performance of either system, but it does no real harm either. As a side effect it makes both systems run slightly faster and, at least with MAWS, produce more compact descriptions, so the effort needed to develop it was not entirely wasted.

8.6 Evaluative Review

This chapter has recounted the development and testing of a suite of programs that constitute a workable text-classification system, called GLADRAGS. This software incorporates some quite venerable ideas but also introduces some potentially valuable innovations to the field of what may be termed textual data mining.

An appraisal of this system, in the context of the project as a whole, will form part of the last chapter of this thesis; meanwhile a more detailed (but less global) interim assessment of some of its main strengths and weaknesses is attempted here.

8.6.1 Achievements

First of all, GLADRAGS is an operational inductive classifier that runs acceptably fast on a 486-based personal computer. Its performance on the numeric data sets (section 8.2) shows that the basic framework is sound. Its performance on the 13 textual benchmark problems is similar to that of the Robust Bayesian classifier of Chapter 6 -- somewhat better in terms of the logarithmic penalty criterion, slightly worse in terms of percentage success rate (see Table 8.23). Above all, it constitutes an existence proof that an evolutionary approach to machine learning can be applied with some success to problems in this area.

In doing so, it makes a contribution to the ongoing search for an effective way of finding meta-features. This is by no means a solved problem: it crops up in several contexts within the field of pattern recognition and machine learning. For instance, the authors of CART (Breiman et al., 1984), state this problem as follows.

> "In some data, the classes are naturally separated by hyperplanes not perpendicular to the coordinate axes. These problems are difficult for the unmodified tree structured procedure and result in large trees as the algorithm attempts to approximate the hyperplanes by multidimensional rectangular regions." (p. 132)

Their tentative solution involves a search for compound features involving linear combinations of features as tests at each node of a classification tree. They admit, however, that their method is non-optimal in two important respects.

> "First, it is still not clear that the present version [of the procedure that searches for feature combinations] consistently gets close to the global

> maximum. Second, the algorithm is CPU-expensive." (Breiman et al., 1984, p. 173)

This procedure has been subject to minor modifications since 1984 (e.g. Murthy et al., 1995), but without materially altering the truth of the above quotation; and without meeting a third objection which can be raised against it and its descendants: namely, that **non-linear** combinations of features may well be important, yet the CART procedure does not even seek such things, and therefore cannot find them.

GLADRAGS cannot pretend to solve once and for all this important threefold problem, but it does confront it head-on, and at least suggests that an evolutionary-computing approach to the discovery of (possibly non-linear) compound features is worthy of further consideration. In other words, the work reported in this chapter throws some light on the delicate question of to what extent it is worth having more complex tests at each node of a discrimination tree or similar structure in order to reduce the overall size of that structure.

Another useful contribution of this system is that it does provide a vehicle for operationalizing and thus testing certain important questions concerning representation and inference in the context of inductive classification, as reported in sections 8.3 and 8.4. This has allowed some interesting findings to emerge. The fact that choice of fitness function can be very important to the success of a rule-based classifier using a genetic search is by no means surprising; but the finding that tabular linkage appears to be the best way of performing Bayesian inference, of three plausible methods tested, could have significant repercussions outside the stylometric domain. The `naive' Bayesian approach has the virtue of simplicity, and the decision list formalism of Rivest (1987) has the virtue of elegance; but neither of them was as effective with the problems tested in this chapter as the signature table concept proposed back in the 1960s by Samuel (1967) during his work on self-improving checker-playing programs. In view of the prevalence of Bayesian reasoning in cases where probabilistic classification is appropriate (surely a majority of classification problems), this constitutes a genuine finding.

Another worthwhile aspect of GLADRAGS is its employment of the heap-tree representation. This would appear to be a viable compromise between the needs of genetic rule-learner to have simple single-level `chromosomes' to chop up and recombine and the preference of human users for a rule language that allows hierarchical relationships among its components. This form of rule representation is (as far as I know) a novelty introduced to the world of machine learning through the present thesis, and it works well enough to deserve wider exposure. It may well prove to be a bridge between the sometimes Procrustean simplicity of traditional GA bitstrings and the somewhat unprincipled representational liberalism of Genetic Programming as advocated by Koza (1992) in particular.

Finally, it is worth mentioning two potentially useful `spin-offs' stimulated by the work of this chapter, although not inherent to the design of the GLADRAGS suite.

The first of these is the introduction of Brier scoring to the practice of comparing classifiers. Percentage success rate is an easily understood criterion, but -- especially in cases with very unequal prior probabilities (e.g. in many medical contexts where over 99% correct decisions can typically be achieved by a rule that always predicts the absence of any particular disease) -- it is also rather a crude one. Brier scores have been used for a decade and a half in studies of human judgement but have not apparently percolated into the field of machine learning. If the present study is a step towards more general acceptance of Brier scores in this field, then it will have performed a valuable service. Brier scoring, using what amounts to a squared-error criterion, appears to be more informative than using success rates while avoiding the over-sensitivity that seems to typify the use of a logarithmic scoring system, where errors attract an exponentially increasing penalty as they become more confident. (It is only a pity that Brier scores were not adopted earlier, and used consistently throughout this project.)

A second spin-off from this chapter is Iota (an Index of Textual Association). This quantity, although clearly not the last word on the subject, is quite serviceable, and will no doubt prove valuable in future attempts at discovering textual (and perhaps non-textual) features automatically. (Having said this does not preclude further refinement of this measure nor the development of other measures that attempt to quantify the three desirable attributes of textual markers: commonness, distinctiveness and stability.)

8.6.2 Problems with GLADRAGS

The main problem with GLADRAGS is that, neither in terms of success rate nor in terms of logarithmic penalty, does it match the performance on unseen data of BBTC, the Basic Bayesian Text Classifier of Chapter 4 -- which was not meant to be a `state-of-the-art' text classification system. Judged harshly, therefore, GLADRAGS -- like MAWS and TEXTREE in Chapter 6 -- is a failure: the objective of creating a system that produces intelligible (rule based) descriptions as well as giving more accurate classification than a simple but opaque method remains unachieved.

It is arguable that this is the most significant finding (or perhaps `non-finding') of the whole project. As Richard Feynman says, in a plea for scientific integrity:

> "If you've made up your mind to test a theory, or you want to explain some idea, you should always decide to publish it whichever way it comes out. If we only publish results of a certain kind, we can make the argument look good. We must publish *both* kinds of results." (Feynman & Leighton, 1992, p. 343.)

Of course this failure to beat BBTC with a `transparent' rule-based classifier might just be a reflection of the present author's competence or otherwise as a programmer[32]; but it could equally well point to some fundamental differences between typical text classification tasks and the kind of problems normally dealt with by statistical classifiers or machine-learning methods. Implications of this sort will be considered further in Chapter 10.

The experiments in this chapter could also be criticized for examining only a small fraction of the fitness and quality functions that might be conceived, which implies that an optimum combination has probably not yet been found. There is also a case to be made that the rule language employed here is not, after all, very well adapted for the task of text classification. Both these weaknesses can, however, be seen from another perspective as opportunities -- opportunities for further research into better fitness functions and more effective representation schemes. They do not necessarily invalidate the general schema instantiated by GLADRAGS.

Another weakness is that both from a software-engineering perspective and in terms of its user interface, GLADRAGS still leaves much to be desired. It is still very much what used to be called a `proof-of-concept' demonstrator during the heady days of the Alvey Programme. Such deficiencies, however, could be cured with time. A related, more significant, problem is the fact that GLADRAGS currently has no module concerned with tidying up a rule base in order to present it in its simplest form for human consumption. In short, GLADRAGS has no post-processing program that applies arithmetic and logical laws to replace, for example, (a+0) by a, (x*1) by x, and so on.

This omission is also curable given time. With short rules, of only 16 bytes, it hardly matters; its seriousness with longer rules can be best judged with the help of an example. Table 8.24 shows the rule base and associated signature table produced by CAPE from one of the most difficult benchmark problems, the FEDS data, with a rule length of 32 and monadic operators absent (i.e. using the favoured parameter combination specified in 8.4.3.)

[32] For what it is worth as evidence: I tried very much harder to make a success of GLADRAGS than of BBTC, expending (I would estimate) about 30 times more work on the former than the latter.

Table 8.24 -- Rule Base from Federalist Training Data.

```
((`tes` * `tat`) + (` constitut` ^ `very `))
  > ((37 * ` upon `) * (23 \ `i`))

((`nd ` - ` of `) \ (32 \ `, an`))
  > ((44 \ ` of t`) \ (`e s` + `es, `))

((` one ` * 92) \ (`an` * `r `))
  > ((59 ^ `nd`) + (47 * `wer`))

((31 + `s, a`) * (`e sta` ^ `power`))
  > ((7 ^ `n`) - (` and ` - ` upon `))

22 markers actually used :
`nd `
` and `
`nd`
` upon `
`an`
`n`
`power`
` of `
`wer`
`e sta`
`, an`
`i`
`e s`
`s, a`
`r `
`tat`
` constitut`
` of t`
` one `
`es, `
`tes`
`very `

Main frequency table, with 16 rows :

        542     4
```

	244	58	240
0000	123	17	22
0001	24	1	18
0010	2	8	0
0011	0	0	0
0100	2	9	0
0101	1	0	2
0110	0	5	0
0111	0	0	0
1000	68	6	83
1001	20	0	109
1010	3	1	3
1011	0	1	0
1100	1	8	0
1101	0	0	3
1110	0	2	0
1111	0	0	0

At first glance, this rule base (which retains some dubious markers including `power' and ` constitut' despite pre-filtering by POSTMARX and selection by TOGA) is not very illuminating. However closer scrutiny does throw up some points that a serious analyst might want to pursue. One of these is that, reading the signature table in conjuction with the rules, it would seem that there is only one reliable rule-combination indicative of Madison's authorship -- 1001, the case when the first and last rules are true and the other two false. A second point is that the null combination, when all four rules are false, contains the majority of Hamilton's examples. This implies that the middle two rules are indicators of Jay's authorship. The rule base is obviously employing a rather dangerous strategy, as it means that Hamilton will be presumed author in the absence of positive signs to the contrary.

Overall, then, it is suggested that although such rule bases are less effective than was hoped, and despite the fact that they do require a certain amount of learning by a human reader before they become truly `transparent', the GLADRAGS system does have some advantages over BBTC (and over alternative classification systems that were not specifically tested during this project but which have been used for similar tasks, such as multi-layer neural nets). To conclude, it may be said that GLADRAGS provides certain sorts of insight into textual classification problems that are simply unavailable in other ways. While leaving plenty of room for improvement, it has pioneered a route worthy of further investigation.

Chapter 9

THE *FEDERALIST* REVISITED (AGAIN) : A CASE STUDY

> "If it became cheap to transfer text from print to computer through, say, a reading machine, the main obstacle to authorship studies would vanish." -- Mosteller & Wallace (1984).

The next chapter will look back and endeavour to arrive at some general conclusions concerning text categorization; meanwhile, as a final forward step, this chapter tackles a particular problem in text discrimination to show how some of the techniques and software developed during the present project might be used in practice.

9.1 A Break from Benchmarking

The preceding five chapters have been chiefly devoted to benchmarking, a principal objective being to find out whether and by how much some methods of text classification work better than others. It is contended that the prominence given to benchmarking in this thesis has certain advantages. Criticisms of the actual choice of the 13 test problems used are easy enough to make (and some will be made in the next chapter) but, between them, they still cover a broader range of problem types than attempted in any previously published stylometric study. Thus the danger of overspecializing methods so that they only work well on one or two problems is largely avoided. Moreover, the inescapable empiricism of such an approach ensures that theoretical elegance -- if it does rear its head -- remains in the service of operational efficacy, and does not become a goal in itself; thus, it is hoped, protecting this work from the accusation of being `merely' academic.

But, of course, this bias towards benchmarking also has disadvantages. In particular, a concentration on comparing techniques tends to give only a patchy idea of how those techniques might actually be used in a case where the object is not to test various systems but to employ them to cast light on a specific problem.

The present chapter aims to redress this imbalance, at least in part, by returning to the *Federalist Papers* and conducting a small-scale case study on them, using some of the software tools developed in earlier chapters.

This celebrated stylistic problem was chosen:

(1) for convenience;
(2) because it forms the basis of one of the 13 benchmark problems and thus gives an idea, at least indirectly, of how other such problems might fare if subjected to more intensive individual examination;
(3) because it connects with many earlier stylometric studies (including Mosteller & Wallace (1964, 1984), McColly & Weier (1983), Merriam (1989), Martindale & McKenzie (1995)) and thus allows external comparisons to be drawn;
(4) because it is widely acknowledged as a severe test.

To expand slightly upon the last point, in chapters 4 to 8 the FEDS problem has consistently emerged as the hardest or second-hardest case (depending on which quality measure is used) and mention has been made that results achieved on it were disappointing, hardly ever exceeding 60% of categorizations correct. However, these results have been obtained while labouring under a number of restrictions that other researchers have not imposed on themselves and which would be unnecessary if using the classification systems to best advantage in order to shed a little more light on the Hamilton-versus-Madison authorship dispute were our main purpose.

In other words, such (self-imposed) restrictions, while advantageous from the point of view of setting up a challenging test, are counter-productive from the point of view of using the software most effectively. The most important of these restrictions are:

(1) working with sonnet-sized chunks -- lines of text of approximately 640 bytes or just over 100 words each;
(2) testing only on text samples whose provenance is in no doubt;
(3) insistence that textual markers be found automatically, without human intervention of any kind;
(4) inclusion of text by John Jay, the third author whose writings form the smallest part of the *Federalist Papers*, thus making a 2-category problem into a 3-category problem.

The inclusion of John Jay seems, in retrospect, self-defeating. It makes the FEDS problem more difficult than it might otherwise be (which is good in one sense) but much of that difficulty arises from relatively uninteresting causes: firstly, the fact that there is not a wide enough selection of Jay's material to be representative in either test or training sets; secondly, the fact that the prior probabilities of Hamilton, Jay and Madison depart rather drastically from equality; thirdly, the mere fact of having an extra category. (Coping with highly unequal priors is not a trivial problem, but it is by no means unique to the area of text categorization and thus finding a means of solving it is only peripherally relevant to the present study.)

An unwillingness to tamper with the output of the Monte-Carlo feature-finders (e.g. CHISUBS and TEFF) is an excellent principle: it avoids confusing the issue of how well the software performs with how ingenious the analyst can be -- which may mask differences between systems when they are the main focus of the inquiry. This principle will therefore be upheld in what follows, even though human editing of a list of textual markers could, in a real application, provide an avenue through which, quite legitimately, domain expertise might be introduced into an analysis. However, it will be interpreted slightly more liberally than hitherto[33]. Specifically, markers found by two different programs will be merged and combined. This is quite easy to arrange by using CHISUBS and STRETCH to produce one list of substrings, TEFF to produced another, then joining these two sets and filtering the combined list through POSTMARX, which discards unstable markers and automatically ignores any duplicates.

The insistence on using only well-attested writings is, again, sound in principle, although here it leads to a relative paucity of sample text by Madison. Mosteller & Wallace (1984) circumvented this by bringing in a large amount of Madisonian text from outside the *Federalist Papers*, mainly from Madison's work on *Neutral Trade*. A less arduous and more audacious solution -- possible only subsequent to Mosteller & Wallace's work -- is to accept the view of Martindale & McKenzie (1995), who state:

> "We began this article with the assumption that Mosteller and Wallace's (1964) conclusion that Madison wrote the disputed *Federalist* papers is so firmly established that we may take it as given."

On this basis, earlier chapters have perhaps been over-scrupulous in confining themselves to Madisonian papers whose authorship was never in doubt even before 1964. The scholarly consensus is now that Madison did write the disputed papers and there is no reason to challenge this view. It is therefore appropriate for a modern re-analyses to proceed on this presumption and see what is revealed by doing so.

[33] Reliance on machine-generated markers has seldom been adhered to as strictly as here. For example, Binongo (1994), who in general follows Burrows (1992) in allowing the texts under scrutiny to dictate what features are used, by employing the most common 36 words as markers, nevertheless says: "all pronoun forms are to be excluded as their occurrence depends to a considerable extent on the context (e.g. some stories do not have female characters)." He further notes that if pronouns are included and the commonest 36 words of the text samples used without selection then performance indices "drop drastically", from 96% to 75% correct in one pairwise comparison between authors and from 98% to 67% in another.

As for using the sonnet-sized chunk as a standard textual unit, this was always an arbitrary decision, at least when applied to prose. It is instructive as an attempt to push the boundary of what is regarded as a suitable unit of text closer to where signal is drowned in noise than previous workers have thought feasible, but the precise size of 640 characters is by no means essential to the techniques and software developed during this project. Indeed it would be interesting to find out some more about how the length of textual unit chosen affects accuracy of classification.

In the following study, therefore:

(1) longer text segments will be used;
(2) the `disputed' papers will be provisionally accepted as Madison's work;
(3) a dual feature-finding search will be used;
(4) John Jay's samples will be excluded as a mere distraction.

Initially, then, the effect of simultaneously relaxing the four main constraints discussed above will be examined by using the three systems that have appeared most promising in the benchmarking exercises of the preceding five chapters: BBTC, the Basic Bayesian Text Classifier; MAWS, the robust Bayesian classifier; and GLADRAGS, the Darwinian rule-generator.

To do this, a fresh division of *Federalist* papers into test and training sets was made. For Madison, all undisputed papers constituted the training sample while the disputed papers constituted his test set. For Hamilton a purely random selection of 17 papers was chosen as training sample with another random selection of 13 papers as test set, giving test and training sets of roughly the same size as Madison's. Paper numbers are listed in table 9.1 below.

Table 9.1 -- Papers Used in this Experiment.

	Paper Numbers
Hamilton, training data (17 papers)	6, 7, 9, 11, 12, 17, 22, 27, 32, 36, 61, 67, 68, 69, 73, 76, 81
Hamilton, test data (13 papers)	1, 13, 16, 21, 29, 30, 31, 34, 35, 60, 65, 75, 85
Madison, training data (14 papers)	10, 14, 37-48
Madison/Disputed, test data (12 papers)	49-58, 62, 63

These texts were combined into four files and the PREP program (see Chapter 4) was run to split them into lines of approximately 1250 characters each. This line length was chosen so that each text unit was about 1/10th of the size of a disputed *Federalist* paper. At the same time this is nearly double the length used in the foregoing benchmark studies.

Details of the size of these samples are given in Table 9.2, below.

Table 9.2 -- Sizes of Text Samples Used.

	Kilobytes, rounded	Words	Lines	Mean Bytes per Line, rounded
Hamilton, training data	225	37921	180	1250
Hamilton, test data	153	25928	122	1253
Madison, training data	234	38710	187	1248
Madison, test data	144	23999	115	1250

The results of running the three systems named above on these text samples are summarized in Table 9.3.

Table 9.3 -- Results, after Training, on Test Data.

System ↓	Markers Used	Success Rate (%)	Log2pen	Brierval	Precall
BBTC (using digrams)	773	81.43	-0.8697	0.5890	0.8140
MAWS	39	83.97	-0.5670	0.7638	0.8396
GLADRAGS	15	78.06	-0.7104	0.6721	0.7808

The last four columns record performance on unseen data, using the same measures as described in Chapter 8. As usual, for the stochastic systems (MAWS and GLADRAGS) figures given are medians of five runs. (BBTC is deterministic, and thus gives identical results when re-run.) In the case of GLADRAGS, the parameter settings recommended in 8.4.1 were used.

For ease of comparison, baseline figures are presented in Table 9.4 below. These have been collated from earlier chapters, with all conditions equivalent to those of Table 9.3 except that the limitations discussed above were in force (i.e. with 640-byte chunks, avoidance of disputed Madison papers, marker substrings from only a single search and inclusion of John Jay's papers as a third category) -- other than Brierval, which has not been quoted previously.

Table 9.4 -- Comparable Baseline Figures from Previous Chapters.

System ↓	Markers Used	Success Rate (%)	Log2pen	Brierval	Precall
BBTC (using digrams)	701	57.57	-1.3825	0.4201	0.4858
MAWS	36	57.82	-1.63	0.3767	0.4573
GLADRAGS	21	53.85	-1.50	0.4031	0.4066

These figures show a very clear improvement, by all three systems, when the four main constraints discussed above are relaxed. MAWS would seem to benefit most: it goes from being worst on two out of four performance measures to being best on all four. Conversely, BBTC benefits least. Even so, in these revised conditions, BBTC attains a highly respectable success rate of more than 81% correct. MAWS consistently does better than that, with a median of almost 84% classifications correct. Furthermore, each of its five trials gave a higher success rate than that achieved by BBTC; and according to logarithmic penalty and Brier scores its superiority is even more pronounced. Far from being `disappointing', MAWS begins to look like a very useful text classifier.

GLADRAGS does less well than the other two systems in terms of success rate and Precall, but its probability estimates are reasonably reliable -- as indicated by the logarithmic penalty and Brier scores. It also clearly gives the most economical descriptions of what it has learnt, as shown by the number of marker substrings retained. Indeed on one of its five runs GLADRAGS kept only a single rule that used just six variables, as shown below.

```
((`rs` ^ `ern`) \ (102 ^ 27))
  > ((` upon` + ` there `) * (`ld` ^ `e t`))
```

Since neither `rs' nor `ern' could conceivably occur as many as 102 times in a single line, this rule can be simplified to

```
(`rs` ^ `ern`)
```

```
  > ((` upon` + ` there `) * (`ld` ^ `e t`))
```

where `^' is the maximum operator (as explained in 8.1.2).

When this rule is true, it indicates Madisonian authorship; when false, it indicates Hamiltonian authorship. It is a very compact characterization of the difference between Madison's and Hamilton's style of writing. It is also quite successful, as shown in Table 9.5, which gives the confusion matrices for this rule on both training and test data.

Table 9.5 -- Confusion Matrices for a Single GLADRAGS Rule.

True Category → Rule Decision ↓	Hamilton	Madison	Unseen Data →	Hamilton	Disputed
Hamilton	129	20		93	23
Madison	51	167		29	92

Here the rule's performance on the unseen data is shown in the last two columns. The great majority (76.23%) of chunks from Hamilton's unseen papers are, correctly, assigned to Hamilton while the vast majority (80%) of disputed lines are assigned to Madison by this rule. It is hard to believe that such divergent behaviour would be exhibited if Hamilton were indeed the author of the disputed papers.

As well as being quite successful, this rule serves as a prompt to further questioning. For instance: why have these 6 markers survived the various selection processes involved? Both ` upon' and ` there ' are well-known Hamilton markers so their position in the right-hand side of the inequality is explicable; but what of the other four substrings? Table 9.6 lists some of the contexts within which these four substrings were found more than once, for both authors.

Table 9.6 -- Contexts within which Marker Substrings Occur.

Substring ↓	Occurs in Hamilton text within:	Occurs in Madison text within:
\`ern'	government(s) governed external internal western	government(s) concerning
\`rs'	wars personal	first personal creditors debtors
\`ld'	would should could gold	would should could hold
\`e t'	be to are the more than	same time able to are too

The substring \`e t' is the most heterogenous of the four. It appears in a wide variety of contexts. A perusal of them suggests, however, that most of them -- for both authors -- conform to the pattern of a verb ending in \`e' followed by a function word, as exemplified in Table 9.7, below. This table is to be interpreted on the understanding that almost any of the example words in the left column can be followed by any of the words on the right to form an instance of the general scheme.

Table 9.7 -- General Pattern of Contexts containing `e t'.

Verb	Function Word
have	to
are	the
be	there
were	they
become	their
make	them
sacrifice	themselves
require	that

The words `to' and `there' are fairly reliable Hamilton badges, but that cannot be the whole story behind `e t', since both those markers are available for selection in their own right. Quite possibly this is the best the system can do, within its current limitations, to exploit an underlying grammatical or syntactic pattern that distinguishes these two authors. At any rate, this sort of finding invites further investigation.

As for `ld', more than half its occurrences occur within the word `would' (with the rest mainly in `could' and `should'). `Would' is a weak Hamilton badge that was in fact eliminated by POSTMARX as insufficiently stable. The appearance of `ld' is thus a kind of rediscovery and generalization of this marker.

Another substring occurring in a wide variety of contexts is `rs'. Although marginally more frequent in Madison's samples than Hamilton's, neither it nor `ern' is really a Madison marker as such. Rather they seem to be functioning as substrings which both authors tend to employ quite often (given that `government' is the central topic of their book) thus providing a background level against which the essentially Hamiltonian subexpression on the right-hand side of the rule may be compared. In other words this rule is treating Hamilton as the more distinctive author, with Madison as a kind of default decision when Hamiltonian characteristics are absent or rare.

So GLADRAGS cannot be summarily dismissed, despite achieving the lowest percentage success rate of these three systems, as it is most likely of the three to provoke further insight into the data.

Overall, then, it is evident that liberation from the `straight-jacket' of benchmarking turns these three systems into useful text-analytic tools. (Perhaps this in itself justifies the benchmarking effort.) But, in removing all four restrictions at once, four different factors have been confounded. It would be interesting to know whether they are all equally important, and, if not, which of them is causing the most improvement.

9.2 A Stepwise Removal of Restrictions

To assess this, each of the four restrictions was removed, one at a time, in the following order.

Table 9.8 -- Codes & Descriptions for the Four Factors.

Code Letter (used in Figs. 9.1 & 9.2)	Brief Description of Factor:
F	Use of Dual Feature-Finding
J	Omission of text by John Jay
D	Acceptance of Disputed papers as Madison's
L	Extension of standard line size from 640 to 1250 characters

To summarize the results, Figure 9.1 shows the success rate attained by all three systems on unseen data as each of these factors is successively altered from its unfavourable to its favourable condition[34]. In this diagram `baseline' refers to the performance on the 3-category FEDS problem used as a benchmark test in previous chapters. Letter codes as in Table 9.8 have been used on the X-axis to designate the other four conditions.

[34] Dual feature-finding cannot benefit BBTC, as, in effect, it finds its own markers (digrams in this case) and does not rely of the Monte-Carlo feature finders. Thus, for this program, the two left-most Y-values in the graph are identical.

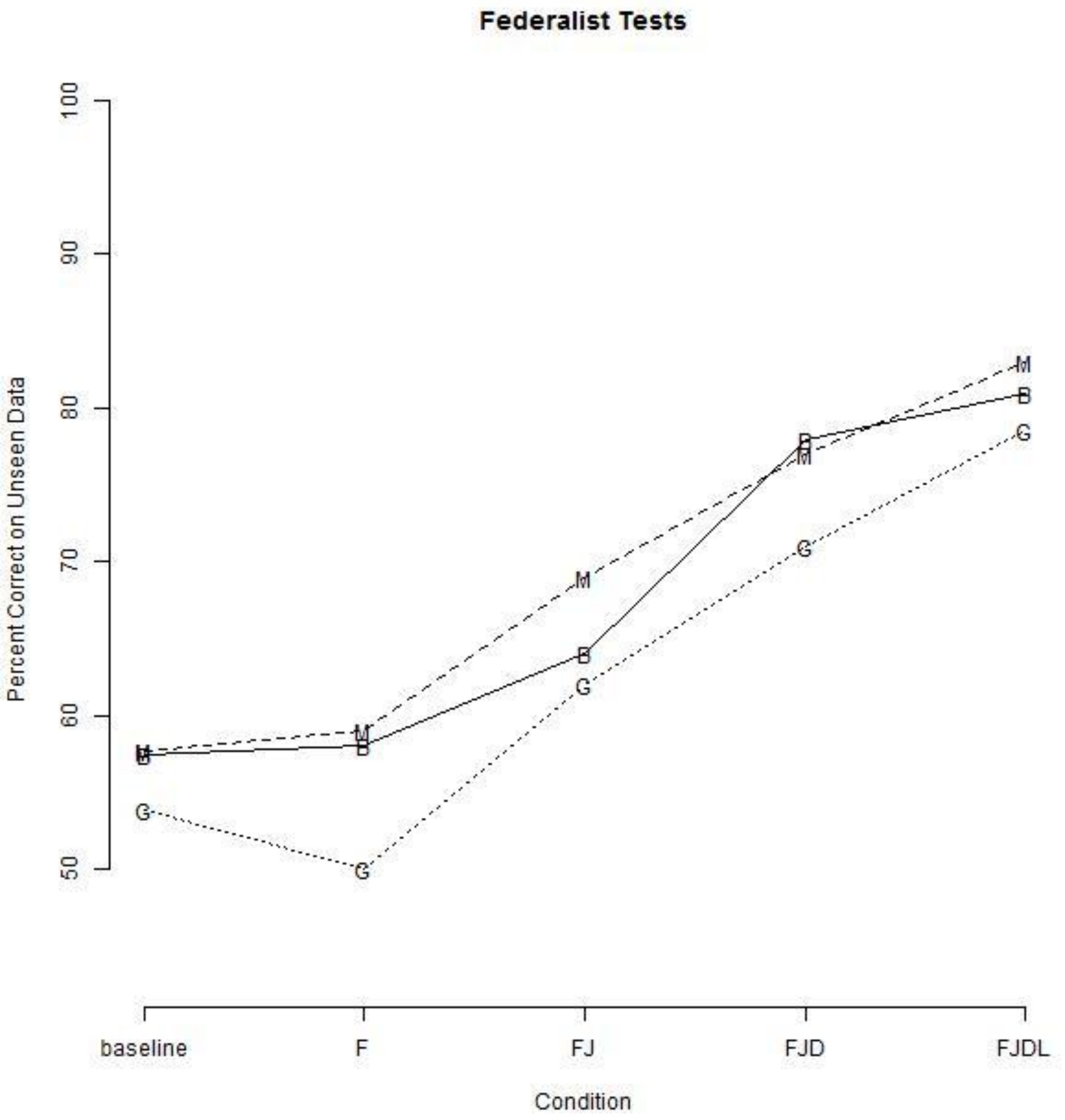


Figure 9.1 -- Success Rates as Restrictions are Removed.
[B = BBTC, M = MAWS, G = GLADRAGS.]

It seems clear from this graph that the effect of dual feature-finding is negligible, but each of the other three relaxations appears to give a worthwhile increment in success rate. Much the same picture emerges using Brier score as a performance measure (probably a more sensitive one, as argued in the last chapter). This is shown in Figure 9.2.

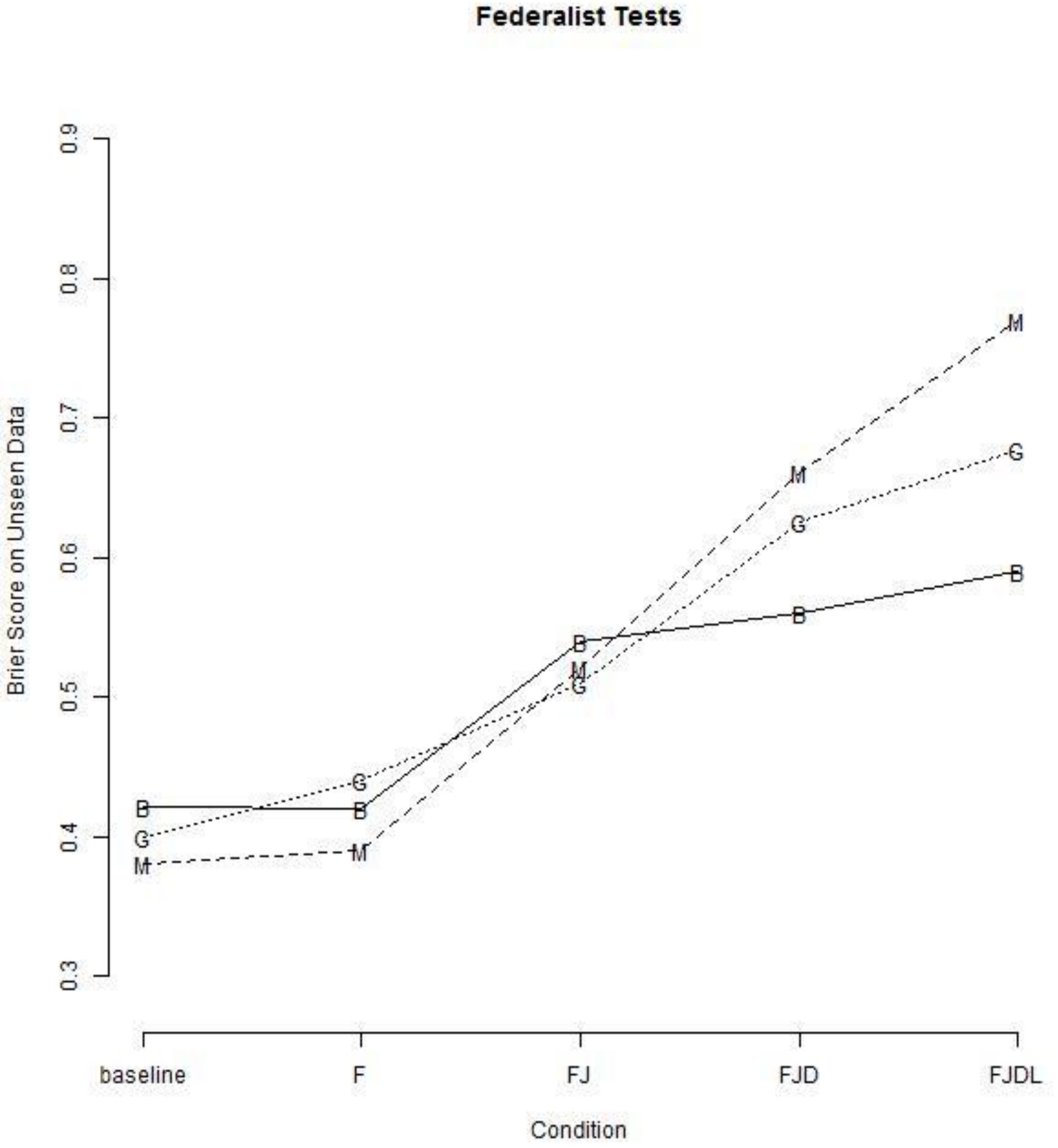


Figure 9.2 -- Mean Brier Scores as Restrictions are Removed. [B=BBTC, M=MAWS, G=GLADRAGS.]

In terms of Brier scores, MAWS is the worst system when all restrictions are in place but clearly the best when they have been removed. It is thus the most brittle, or, to put it more charitably, the system with greatest potential. These results can be translated into advice on using the systems effectively:

(1) try to stick, if possible, to 2-category discrimination problems;

(2) collect a reasonably large sample of training data (as the difference between conditions FJ and FJD is most likely attributable largely to an increase in the sizes of the training sets from 154 and 151 kilobytes of text to 225 and 234 kilobytes);

(3) use a block size (i.e. standard line length) of over a kilobyte where possible.

These suggestions apply to all three systems, but to MAWS with most force. Taking them into account, it would seem that a mostly Bayesian approach to probabilistic text classification that depends on using short substrings as distinctive markers can be made to work well on a difficult problem.

9.3 Though this be Madison, Yet there's Method in it

This is all very fine as far as the utility of the software is concerned, but does it tell us whether Madison wrote the disputed *Federalist* papers?

Table 9.5 is hard to interpret in any other way than as strong evidence against Hamiltonian authorship of the disputed papers; and this was obtained from the system that gave the weakest result of the three. However, it applies in aggregate, to the whole body of disputed papers, not to each paper individually.

Division of texts into lines of a standard length, as practised here, causes loss of information about where a particular work (poem or, in this case, essay) begins or ends. However, as the line size chosen was approximately 1/10th of a typical *Federalist* paper, some further information can be gained simply by taking decisions made by the classifiers on individual lines 10 at a time. This has been done in Table 9.9, which gives results obtained on 10-line groups of the Disputed papers by all three systems.

Table 9.9 -- Number of Madison Decisions in Groups of 10 Lines.

10-line segment (c. 2100 words each)	BBTC	GLADRAGS	MAWS
1.	8	8	7
2.	10	9	9
3.	9	9	8
4.	8	10	8
5.	8	8	6
6.	9	7	9
7.	7	6	9
8.	8	8	8
9.	7	9	8
10.	6	10	6
11.	7	7	8
(12.)	(6)	(2)	(10)

For the stochastic systems (GLADRAGS and MAWS) these counts have been taken from whichever run gave the median success rate quoted in Table 9.3. The parentheses in the last row of this table reflect that fact that the PREP program actually divided this set of 12 papers into 115 lines (not 120), so the final count is based on five lines (and doubled, for comparability), rather than on 10 as with all the rest.

What is striking about these figures is that in every case but one, all three classifiers give a **clear majority** (with no ties) of lines to Madison. The single exception is with GLADRAGS on the incomplete final chunk (from the tail end of paper number 63). Taking a consensus of the three systems by adding the last three columns together, even this `mistake' disappears; and for the 11 full-sized segments the lowest total is 22 out of 30. This suggests that a success rate of over 80% on 1250-byte or 210-word chunks translates into virtually 100% success rate at the level of a 2100-word paper -- provided that a suitable method of aggregation can be found. (At present such aggregation must be done by hand, so more work is called for to make this automatic, and perhaps more sophisticated than a simple tally[35].)

[35] Within a generally Bayesian framework, the most straightforward way of combining evidence from a number of text lines would be to multiply likelihoods

Hamilton's putative authorship of these lines seems an untenable hypothesis, bearing in mind the fact that all three programs were approximately equally decisive in classifying the genuinely Hamiltonian unseen material as Hamilton's.

Further evidence can be adduced from running these programs `backwards', so to speak, i.e. training them on the test sets (using therefore the disputed papers as Madisonian exemplars) and testing them on what up till this point have been considered the training data. The results of doing so are given in Table 9.10.

Table 9.10 -- Results when Test & Training Sets are Exchanged.

System ↓	Markers Kept	Success Rate (%)	Log2pen	Brierval	Precall
BBTC	672	64.85	-0.8882	0.5742	0.6504
MAWS	41	76.02	-1.3541	0.5857	0.7613
GLADRAGS	20	67.85	-1.0022	0.5264	0.6799

These are considerably worse results than obtained in the normal or forward order, showing that similarity as assessed by these programs need not be symmetrical (and also showing that logarithmic penalty is the most volatile of the quality measures). However, much of this drop in performance can be explained by the fact that the training samples are now smaller (down from 225 and 234 kilobytes to 153 and 144 kilobytes). A slightly lesser size differential was associated with a degradation of about 7.8% in success rate for MAWS, as discussed in the previous section.

In any case the results from MAWS still lend weight to the contention that Madison wrote the disputed papers. Table 9.11, below, gives the confusion matrix of MAWS on the undisputed Hamilton and Madison data when trained on Hamilton's and the disputed texts. This result comes from the run that produced the median success rate of five.

from each line, then multiply these products by the relevant priors and rescale to sum to 1. However, this ignores the sequential structure of text. Successive chunks of text will almost certainly be more similar than randomly chosen chunks; in the language of time-series analysis, there will be serial autocorrelation. Unless some means of dealing with this is found, a simple-minded application of Bayes's Rule will tend to give exaggerated probability estimates.

Table 9.11 -- Confusion Matrix from MAWS, Reverse Trained.

True Category → Program Decision ↓	Hamilton (unseen)	Madison (unseen)
Hamilton	148	56
Disputed (Madison)	32	131

Thus the system, when trained to distinguish Hamiltonian from Disputed lines of text, classifies over 70% of Madison's (previously unseen) lines as Disputed while allocating over 82% of the genuine but unseen Hamiltonian lines to Hamilton. It is extremely hard to reconcile this evidence with a belief in Hamilton's authorship of the disputed papers.

To the extent that this software is effective, these results tend to reduce slightly any lingering doubts about Madison's authorship of the disputed *Federalist* papers. Hopping onto the other foot (as it were): to the extent that Madison's authorship is well established, these results tend to confirm the effectiveness of the three systems employed here as tools of textual analysis.

Given that this software can be useful in authorship studies or similar text-classification tasks, it is perhaps as well to give explicitly an outline of a methodical way of using it.

(1) Collect plenty of undisputed text of the types (preferably only 2) under scrutiny -- perhaps as much as half a megabyte in a hard problem.
(2) Split texts into training and test sets, taking care not to break up single works.
(3) Use PREP to divide test and training data into chunks of roughly equal size, ideally at least a kilobyte each. Mild preprocessing, such as case folding and compacting runs of white space into single blank characters, will normally also be done at this stage. (More aggressive preprocessing, such as eliminating punctuation, may also be done if there is good reason for it.)
(4) Execute one or more of the Monte-Carlo feature-finding programs to create a list of distinctive textual markers. TEFF followed by POSTMARX is probably the best choice here. (Combining substring lists from separate runs does not seem to be necessary.)
(4a) At this point editing of the marker list may be contemplated (and undertaken, provided that it can be justified on publicly understood criteria).
(5) Run BBTC (because it is fast, simple and often quite accurate), MAWS (because it is probably the best system of the three overall), and GLADRAGS (in the hope of finding out more about the data). It is probably wise to run them backwards, from test to training data, as well as forwards from training to test data.

(6) If success rates and Brier scores are sufficiently high[36], continue; otherwise stop (and try another tack).
(7) Examine the substrings kept by MAW1 and the rules retained by CAPE for further clues.
(8) Re-train all three systems on the entire corpus (test and training sets together).
(9) Test on genuinely unknown samples. The estimated probabilities may be taken quite seriously. (Some form of informal aggregation of evidence from the three different systems may well be needed here, given the current state of these systems.)

At present the software is still rather immature, so the above hints cannot be dignified with the name <u>methodology</u>. They are offered as guidelines.

(As already noted, this approach is not meant to preclude other forms of analysis; with a non-trivial problem it is expected that other statistical or machine-learning analyses will also be conducted in parallel with the above steps.)

9.4 Summing Up

To recapitulate, three of the more promising systems arising from this project have been put through their paces in this chapter on a noteworthy and rather difficult problem in text classification -- and have emerged with some credit. To that extent the three main strands running through this thesis

(1) use of short substrings as discriminatory features;
(2) employment of Bayesian inference as the main way of dealing with uncertainty;
(3) reliance on evolutionary computing when it is necessary to search through large solution spaces;

have proved their worth.

It bears repeating that none of the three systems used in this chapter is very close to the state in which they could be `shrink-wrapped' and shipped as commercial packages, or even made generally available as fool-proof public-domain software tools. Nevertheless it is perhaps worth taking them further towards that state, since

[36] Inevitably this is a vague concept, partly dependent on human judgement. If a more conventional analysis is being or has been done on the same data, e.g. using a linear discriminant function or a classification tree (which is advisable in a serious application), then this requirement becomes rather less fuzzy.

this chapter has shown that the methods they embody do have something to offer the field of computational stylometry (and to text analysis in general), especially when used together with their results interpreted conjointly.

With regard to the `disputed' *Federalist* essays, it seems fairly clear that Hamilton, writing just before a duel that was to cost his life, had pressing distractions on (or rather, upon) his mind. In the circumstances, it is all too understandable that he should have written 37-48 instead of 37-58 (a slip of only one digit) and forgotten about papers 62 and 63 altogether.

Chapter 10

CONCLUDING DISCUSSION

"The text unveils itself before us, but never allows itself to be possessed." -- David Lodge (as Morris Zapp, 1984).

This chapter assesses the work undertaken for the present project in the context of the aims stated in Chapters 1 & 2 of this thesis. The evaluation is organized, for ease of exposition, under four headings:

(1) Achievements;
(2) Findings;
(3) Failings;
(4) Future Prospects;

although there is inevitably some overlap between these topics. In particular, failings often point directly to future prospects.

10.1 Achievements

The main aims of this project were stated in section 1.2. These were expanded somewhat, in the light of a survey of previous related work, in subsection 2.4.3. They are briefly summarized below.

An overarching goal was to contribute to the discipline of text analysis by promoting cross-fertilization between the fields of machine learning and stylometry. The means chosen to do this was the development of a generic, automatic procedure for text categorization. In fact, several such procedures were tested and compared. This high-level goal spawned six more specific subgoals, as listed in 2.4.3, which are restated here. (Certain desiderata, that qualify these subgoals but are not themselves aims or objectives, are discussed in section 10.4.)

Main Goal

(0) Develop and test workable text-classification software that:

Subgoals

(1) can deal with more than 2 categories of text;

(2) will find textual markers or features without having to be given them;
(3) uses markers other than words where appropriate, both below the word level (e.g. affixes) and above (e.g. collocations);
(4) draws on a wider range of modern machine-learning techniques than used hitherto in stylometry;
(5) uses statistical techniques such as cross-validation to address the issue of overfitting (i.e. over-optimistic error-rate estimation);
(6) is able to deal with smaller text segments than most previous methods.

The overall goal was indeed achieved: not just one but five trainable text-classification systems were developed and tested, of which three showed considerable promise, namely BBTC (Chapter 4), MAWS (Chapter 6) and GLADRAGS (Chapter 8). No directly comparable study has ever been conducted, but where points of contact exist with previous stylometric research or with other machine-learning benchmark studies, the evidence suggests that the three systems named above are competitive with alternatives being used by other workers. For instance, when GLADRAGS was used on numeric data in Chapter 8 it performed as well as or better than nearest-neighbour classification or linear discriminant analysis; when MAWS was compared in Chapter 6 with TEXTREE, a system using the popular discrimination-tree method of induction (with pruning), MAWS performed better on the text Benchmark suite. Furthermore, when used on the renowned *Federalist* problem in Chapter 9 it performed at least as well as most previous methods reported in the literature.

Subgoal (1) was also achieved, although the effect of removing John Jay from the FEDS problem in Chapter 9 was sufficiently pronounced to suggest that multi-category classification is not a strong point of any of the systems developed here.

The rest of these subgoals deserve somewhat fuller consideration and will be dealt with in more depth in the following subsections (apart from subgoal (5) which was never achieved, and is thus discussed in section 10.3).

10.1.1 The Merits of Monte-Carlo Feature-Finding

Subgoals (2) and (3) were achieved by the development of the technique of `Monte-Carlo Feature-Finding' (MCFF). This technique originated with an attempt in Chapter 3 to rank short substrings according to their distinctiveness for classification purposes. It was extended and refined in subsequent chapters by means of the programs CHIMARX (Chapter 5), CHISUBS (Chapter 6), TEFF (Chapter 7) and POSTMARX (Chapter 8).

Despite being conceptually simple, this form of feature-finding, which relies on short substrings, has proved its effectiveness. For example, in Chapter 5, the nearest-neighbour classifier was more successful using codebook substrings ranked by CHIMARX than using the 100 most frequent words. (It should be recalled that using the most frequent words as features is an approach that has been advocated and used with some success by Burrows (1992) and Binongo (1994), among others.) Also in Chapter 5, the IOGA program, given an equal number of common words and CHIMARX substrings to choose from, chose more substrings than words; and relatively neglected Brunet's W -- a measure of vocabulary richness which has been used successfully as a stylometric feature elsewhere (Holmes & Forsyth, 1995). Thus short substrings, as found by this process, have been shown to be competitive with more traditional stylometric indicators.

Moreover, the great advantage of this sort of feature-finding is its almost complete lack of preconceptions: to a larger degree than usual, it lets the data "speak for themselves". In most other systems of machine learning or statistical pattern-recognition, the choice of features is a burden placed entirely on the user. The learning system or trainable classifier merely coordinates them in some way. Even what is sometimes termed `feature extraction' (e.g. Xu & Yuille, 1992) is not really the same thing: it is more properly seen as a method of dimensionality reduction, as it is reliant on a pre-processing step that expresses the data as a vector of measurements. This is generally true in stylometry as well. Even work that follows the example of Burrows (till now the least prescriptive sort of stylometry in terms of feature selection) is not completely free from user intervention: for instance, Burrows & Craig (1994) in their study of Renaissance and Romantic drama were required to distinguish manually between different syntactic functions of various common words, such as `to'-infinitive from `to'-preposition and `that'-demonstrative from `that'-conjunction; while Binongo (1994), as mentioned in Chapter 9, excised pronouns from the list of common words (and would have obtained materially worse results if he had not done so).

Of course, the idea of a pattern recognizer that finds features for itself is not unprecedented. For example, work in the domain of image recognition using random n-tuples -- initiated by Bledsoe & Browning (1959) and further developed by the work of Steck (1962), Aleksander & Stonham (1979) and Penny (1993), among others -- can be seen as a similar approach to adaptive pattern recognition, albeit without any attempt to economize by selecting a useful subset from among the random n-tuples. Likewise the pioneering work of Selfridge (1959) and Uhr & Vossler (1966), again in the area of image processing -- in which `demons' or `operators' that detect specific features within an image grid are evaluated and periodically replaced by new ones if they prove ineffectual -- can be seen as foreshadowing the present method.

Nevertheless feature-finding from data with minimal preconceptions remains very much a minority tradition within machine learning, and the form of MCFF used here distinctively advances that tradition, which has never been much in evidence in text-processing. In particular, the Iota measure developed in Chapter 8, though clearly not the last word on this subject, offers a practical way of screening out "dangerously contextual" substrings as authorial indicators purely on the basis of their pattern of occurrence within a sample of text -- without recourse to prior sources of knowledge such as a user's intuition or a lexicon or thesaurus. Thus, although much remains to be done, the approach presented here represents an innovation not just to stylometry but to pattern recognition in general[37].

10.1.2 Techniques of Text Categorization

Subgoal (4) was achieved by writing programs for several different methods of classification, applied principally to textual data. Some of these were well-established (e.g. nearest-neighbour classification, discrimination-tree induction) and served to connect this study with other studies, while some were more esoteric (e.g. cross-codebook compression or the GLADRAGS system). Altogether seven different methods were tried. They are listed in Table 10.1.

[37] Although alteration of substrings found by the MCFF programs has been eschewed here for testing purposes, it should be noted that this approach does not preclude user-editing of lists of such substrings where appropriate.

Table 10.1 -- Methods of Text Classification Examined.

System Name	Description	Chapter
DIGS	Classification by Cross-Codebook Compression	3
BBTC	Basic Bayesian Text Classifier	4
1-NNC	Single Nearest-Neighbour Classification	5
IOGA	Instance-Oriented Genetic Algorithm	5
MAWS	Mosteller And Wallace System (Robust Bayesian Categorization)	6
TEXTREE	Textual Discrimination Tree Induction & Pruning Software	6
GLADRAGS	Genetic Learning Algorithms Discovering Relational Algebraic Grouping Signatures	8

Thus, rather than just taking a single off-the-shelf learning method and adapting it to deal with textual data, an attempt was made to find out what sort of adaptive techniques are best suited to text categorization. Much of the work of this thesis consisted of comparisons among these various techniques using a suite of benchmark problems (itself evaluated in 10.1.4 and 10.3.2). Taking, for the sake of argument, results on this suite as indicative of the quality of these classifiers, the main results of chapters 4 to 8 can be brought together and briefly summarized by Table 10.2.

Table 10.2 -- Mean Scores on Benchmark Problems.

Method	**Percentage Success Rate**	**Logarithmic Penalty**
'Default'	47.47	(-1.33)
BBTC (Chapter 4):		
characters	71.99	-1.12
digrams	77.31	-0.88
trigrams	76.68	-0.86
words	67.27	-1.01
digraphs	68.83	-1.07
trigraphs	56.41	-1.56
Euclidean **1-NNC** (Chapter 5):		
codebook substrings	66.01	-
100 commonest words	61.30	-
BOTH (see section 5.1)	66.83	-
IOGA (Chapter 5):		
BOTH	68.37	-
TEXTREE (Chapter 6):		
codebook substrings	63.29	-1.15
Monte-Carlo substrings	66.76	-1.16
MAWS (Chapters 6 & 7):		
codebook substrings	70.16	-1.26
Monte-Carlo substrings	74.10	-1.20
'stretched' substrings	73.76	-1.25
GLADRAGS (Chapter 8):		
extended substrings	70.61	-1.03

Here the `default' method consists of assigning all unseen cases to the commonest category in the training sample. The measures quoted are all derived from unseen test data: they are percentage success rate and `logarithmic penalty'. The latter is calculated from the logarithm, to base 2, of the estimated probability of the actual category. This provides an assessment of probabilistic accuracy. It is explained in sections 4.4 and 8.3.

Although this summary table omits many qualifications and caveats (detailed in the chapters concerned) it still provides a rough indication of the quality of each method of text categorization. It also provides some evidence for the usefulness of Monte-Carlo substrings, i.e. substrings discovered by the Monte-Carlo feature-finding process.

The presence of two standard techniques, 1-NNC and TEXTREE (based on CART), also allows tentative inferences to be drawn about the worth of the more novel techniques developed as part of this project. What is noteworthy is that neither of these standard methods performs particularly well. This is better exposed (at the cost of even more simplification) in Table 10.3, below, in which each of the seven substantively different techniques is represented by its best result, i.e. the condition giving the highest percentage success rate obtained with that technique.

Table 10.3 -- `Pecking Order' of Text Classification Methods Tested.

Technique	Success Rate (%)
BBTC with digrams	77.31
MAWS	74.10
GLADRAGS	70.61
BBTC with digraphs	68.83
IOGA	68.37
1-NNC	66.83
TEXTREE	66.76
`Default'	47.45

These very highly summarized figures are presented as an attempt to encapsulate the gist of a long and multi-faceted study and, being thus condensed, should not be over-emphasized. They should perhaps be read more to provoke questions than to supply answers. Nevertheless it seems fair to deduce from this ranking that the novel methods developed for this project (e.g. MAWS and GLADRAGS) are no worse at text classification than more orthodox methods, such as 1-NNC and TEXTREE. Alternatively, since Michie et al. (1994) have established that nearest-neighbour classification and discrimination tree induction are by no means `straw men', this table can be taken to show that the 13 benchmark problems used for most of the testing in chapters 4 to 8 of this thesis are quite difficult for current discrimination algorithms.

In any case the attainment of subgoal (4) has produced a wealth of specific findings, as well as a general piece of advice: if you want to classify texts, don't assume that simply adapting a standard classification algorithm (whether neural, statistical or inductive) to deal with character strings will result in an optimal text classifier.

10.1.3 Size Does Matter

Subgoal (6) was to produce a text classifier able to categorize smaller segments of text than most earlier researchers have considered feasible. This may seem a mundane, and hence even trivial, objective, but if achieved would have important practical consequences.

Table 2.10 (in subsection 2.4.3) has already indicated that current methods only work with rather large text samples. That this is an enduring problem is confirmed by the quotations below.

> "It is clear in the present study that there is a considerable loss in discriminatory power when samples fall below 500 words". (Baillie, 1974)

> "We do not think it likely that authorship characteristics would be strongly apparent at levels below say 500 words, or approximately 2500 letters. Even using 500 word samples we should anticipate a great deal of unevenness, and that expectation is confirmed by these results." (Ledger & Merriam, 1994)

> "Sampling and a test run using 1,000 words per story were attempted, but the results were not very encouraging, suggesting that 1,000 words are not enough to capture those subtle authorial idiosyncrasies." (Binongo, 1994)

Thus the ability of several of the methods developed here to classify sonnet-sized chunks (of only 640 bytes or around 110 words) with a high degree of accuracy in most of the benchmark problems must count as progress. Even the difficult Federalist problem was eventually tackled with success, in Chapter 9, using segments of approximately 1250 bytes (around 208 words on average).

The fact that relatively short segments of text can be classified should prove a stimulus to further work on the precise relationship between size of textual unit and discriminability.

10.1.4 Incidental Accomplishments

Predictably, some of the more interesting achievements of this project were not on the original list of aims.

Foremost among these, perhaps, is the text benchmark suite itself. No such thing has previously existed, which has meant, roughly speaking, that each stylometer has chosen a single favoured method and applied it, usually, to a single problem -- making it very hard to assess the relative efficacy of different approaches and techniques. Clearly the present suite has many defects, but, with amendments (some of which will be suggested in section 10.3), it or its descendants could become a valuable resource. Above all, the idea (previously alien) that novel techniques need to be proved by trials on certain standard test problems has been introduced into this field. If widely accepted, this should facilitate meaningful comparisons between competing methods and thereby help progress towards more effective text categorization.

Another useful introduction both to stylometry and indeed to machine learning, is the notion that classification performance can and should be measured in ways other than the ubiquitous percentage success rate. Success rate (or, equivalently, error rate) is easy to compute and easy to understand, but it is not the only measure by which a classifier may be judged, and for some purposes quantities such as Brier Score, Logarithmic Penalty or the Spherical Scoring function give a more informative indication of quality (see section 8.3). As Hand (1994) puts it: "performance is typically measured by the misclassification rate. But a simple count of errors leaves much unsaid." On some occasions in this investigation it has turned out that the choice of quality score (e.g. Logarithmic Penalty versus Success Rate) dictates the rank ordering of different classifiers. This suggests that a single index of classification performance can sometimes be misleading and that more work needs to be done on `meta-assessment', i.e. assessing the merits of various ways of evaluating probabilistic classifiers. Although Logarithmic Penalty would seem to have strong theoretical claims on the grounds of its intimate link with Information Theory, experience gained during this project suggests that Brier scoring may well give better results in practice. This thesis can claim to have introduced Brier scoring to the machine-learning community and although it was not used early or systematically enough throughout, it will have been a useful pioneering effort if it leads others to adopt and refine a more thorough-going multi-factorial approach to classifier assessment.

Another useful outcome of this work has been the development of an efficient incremental version of the GA or genetic algorithm (see Chapter 5 and Appendix C). As genetic algorithms were not the primary focus of this research, this procedure was not itself thoroughly benchmarked, but results with IOGA, MAWS and GLADRAGS, as well as unpublished testing on a problem that Jennison & Sheehan (1993) reported to give difficulty to a standard GA, suggest that it may well be one of the more efficient variants of the GA currently in circulation.

In any case, it permitted the development of IOGA, a program that extends the nearest-neighbour classification method by simultaneously selecting a subset of both exemplars and features -- thus discovering a list of archetypes which can be used both for faster classification than the basic 1-NNC and as a more compact description of the structure of the problem domain. This would seem to represent a genuine contribution to the field of instance-based learning, as a thorough literature search has not revealed another system capable of selecting both instances and features at the same time.

Also valuable is the heap-tree rule representation scheme introduced in Chapter 8 for the GLADRAGS programs. This achieves a balance between the rigidity of traditional GA representations (Holland et al., 1986; Goldberg, 1989) and the laxity of genetic programming (Koza, 1992; Kinnear, 1994).

10.2 Findings

Many specific results have been discussed in detail in the preceeding 200 pages. Spacetime limitations permit only a few of the more noteworthy or global to be reviewed in this section.

10.2.1 In Praise of Semi-Crude Bayesianism

Possibly the main finding is a `non-finding', the fact that none of the more sophisticated learning schemes beat BBTC (the Basic Bayesian Text Classifier) on the benchmark suite either in terms of success rate or logarithmic penalty score. BBTC was originally intended rather like a placebo in a clinical trial. It was written quickly, with simplicity as a keynote, mainly to establish a baseline performance level. That performance level was never beaten, despite the expenditure of a great deal of time and effort.

In one sense this is quite embarrassing, but it does not belong with the failings in the next section because it is an outcome with some rather interesting implications. Clarke (1995) has suggested that this case parallels in some respects the research of

Axelrod on the Prisoner's Dilemma game (Axelrod & Hamilton, 1981; Axelrod, 1984). Briefly, Axelrod invited various interested parties to submit programs to play a well-defined adversarial game in a contest arranged so that each contestant (a computer program) was matched with every other contestant and ranked according to the aggregate score it received. This game has been studied quite extensively in psychology and political science and was well known to all participants. In addition, the rules under which the contest would be played were clearly laid down in advance. The first contest attracted 14 entries. It was won by the simplest entry, called TIT-FOR-TAT, submitted by Anatol Rapaport. The strategy used by TIT-FOR-TAT was simply: cooperate on the first move, then do the same as your opponent's last move. Axelrod published full results of this tournament and then organized a second round, which attracted 62 entries from a variety of specialists in psychology, computer science, game theory, political science and allied fields. Some of these entries were very sophisticated, but once again TIT-FOR-TAT came top overall. Of course the present case is not a public competition, but it has some analogical features, of which the main one is that the simplest program does best.

Why should this be so? It is impossible at this stage to give a definitive answer, but a look back at table 10.3 suggests that the use of Bayesian reasoning may have something to do with it. The top three programs in that table (BBTC, MAWS and GLADRAGS) all employ Bayesian inference. The bottom three (TEXTREE, 1-NNC and IOGA) do not. Indeed the overall `winner', namely BBTC, uses the crudest form of Bayesian reasoning, termed naive linkage in Chapter 8, even though this is clearly non-optimal, as the experiments of that chapter demonstrated. These results tend to vindicate Mosteller and Wallace's advocacy of a Bayesian approach in this area, and also to suggest that subsequent workers have been somewhat remiss in neglecting Bayesian methods almost entirely.

However, Bayesian reasoning alone cannot be the whole story, because BBTC achieves a higher success rate than MAWS (which uses the same form of inference) as well as a significantly better logarithmic penalty score. The difference between these systems which would seem on the face of it to offer the most plausible explanation for the observed difference in results between them is the number of features that they employ. With the text benchmark problems BBTC regularly uses over 600 digrams, usually more than 700, while MAWS restricts itself to an average of 25 or 26 substring-features (as it is designed to do). In fact all the other programs (even 1-NNC) employ far fewer features than BBTC. Economy in terms of the number of features used to make categorizations is a desirable, and sought-after, attribute of a classification system; but it begins to seem from these results that to classify texts, especially short stretches of text, it works against high accuracy.

It is characteristic of text that even relatively common features (whether words, subwords or collocations) occur rather infrequently. This effect is more pronounced in short segments, as used in most of the tests reported in this thesis. Given this fact, the superiority of BBTC becomes quite explicable. Its digram table constitutes, in knowledge-engineering terms, a `flat' and rather obscure but extensive knowledge base, better suited to the exigencies of textual discrimination than the more succinct representations of systems like GLADRAGS or TEXTREE. In other words BBTC develops a wider, if more superficial, knowledge base than either GLADRAGS or TEXTREE -- which fits the task in hand better.

These considerations point to a significant mismatch between the techniques typically used in machine learning with numeric feature vectors and the demands of successful text categorization. Goldberg (1995) makes a similar point.

> "The statistical problems associated with sets of document features, noise, high dimensionality and skewed distribution, tend to favor representing concepts that are more disjunctively structured as opposed to conjunctively structured. For example, some learning approaches assume that the concept to be learned can be expressed as a single, conjunctive rule (e.g. Mitchell's candidate elimination algorithm). In text categorization this kind of approach is not likely to work. Because of skewed distribution, most features are not expected to occur in most documents, so a conjunction of features is really unlikely to occur in any given large subset of documents (e.g. all positive examples of a category/concept). All together, these statistical properties suggest the domain of natural language text represented in terms of word-based document features may be highly disjunctive. But, it is a generally held notion that existing concept learning approaches are fundamentally limited in their ability to learn highly disjunctive concepts."

Though perhaps not conclusive, the concurrence of the results obtained here with Goldberg's argument[38] certainly justifies a message of caution to future workers applying machine learning to textual problems. Text makes certain specific demands, and these need to be accommodated.

(Some implications of this finding for the quest to find `transparent' stylograms are considered in 10.3.1, below.)

[38] Strictly speaking, BBTC does not employ a disjunctive representation as it does not employ propositional logic to represent concepts. However in relying on many indicators, most of which seldom occur, it comes closer to the disjunctive than the conjunctive pole of Goldberg's polarity.

10.2.2 The Discreet Charm of Diagnostic Digrams

The relative success of BBTC also reinforces the position of digrams (character pairs) as simple but effective textual markers.

Results from Chapter 4 (recapitulated in Table 10.2) show that digrams are better indicators than single characters, and also that trigrams have no clear advantage over digrams. (Nor in fact do tetragrams or n-grams with n greater than four.) In addition, digrams were found to give better results than words or digraphs.

These findings agree with a number of other studies in several fields. For example, Adamson & Boreham (1974) used a technique based on matching digrams to group document titles into semantic clusters; while Kjell (1994), as mentioned in section 4.7, reported good results using letter-pairs as input to a trainable neural classifier.

The humble digram may not be very exciting from a linguistic point of view, since the most interesting entities in language span much more than 2 characters, but it cannot be overlooked for that reason. In text categorization, digrams are surprisingly effective at picking up quite subtle differences in both content and style.

10.2.3 An Improvement over Current Practice?

Another finding with potentially important implications was the superiority of Tabular Linkage over Naive or Sequential Linkage within a Bayesian inferential framework, as reported in Chapter 8. Bearing in mind that most practical Bayesian reasoning schemes have adopted what is called here Naive Linkage, this result needs to be taken seriously. Unfortunately Tabular Linkage demands more space than the other methods, which tends to suggest that efficient ways of turning full contingency tables into sparse ones with minimal loss of information, such as proposed by Gammerman & Thatcher (1991), warrant further investigation.

10.2.4 Towards a Tentative Taxonomy

A point not previously discussed, which emerged from the benchmarking experiments, was the impression made upon this author that content-based text categorization is easier in general than authorship attribution, which in turn is easier than chronological classification of works by a single writer. This impression was retrospectively investigated by averaging the success rates on unseen test data from the same seven runs of six different systems as listed in Table 10.3 (namely, BBTC with digrams, BBTC with digraphs, 1-NNC, IOGA, MAWS, TEXTREE and

GLADRAGS) on eight of the benchmark problems, subdivided according to the type of problem -- authorship, chronology or content. The results are tabulated in Table 10.4, below[39].

Table 10.4 -- Benchmark Success Rates Tabulated by Text Type.

Problem Name	Mean Percentage Success Rate (from 7 runs)	\`Default' or Expected Success Rate (%)	Differential
LIT1	58.71	39.09	19.62
LIT2	69.57	37.29	32.28
LIT3	60.71	27.17	33.00
	Mean of above 3 Authorship Problems = 63.00		
AGE1	60.71	60.49	0.22
AGE2	76.14	66.67	9.47
AGE3	69.43	66.67	2.76
AGE4	65.14	56.00	9.14
	Mean of above 4 Chronology problems = 67.86		
MAGS	83.14	40.17	42.97

The overall mean for the four chronology problems is actually higher than that for the three authorship problems, contradicting the ordering asserted above

CONTENT > AUTHORSHIP > CHRONOLOGY

but this ignores the fact that in LIT1 there were four categories of text and in LIT2 and LIT3 three categories, whereas in all four AGE problems there were only two, making the former inherently more difficult. To counteract this inherent bias, the last column in the table, labelled Differential, gives the result obtained by subtracting the \`default' success rate, i.e. the rate that would be obtained by guessing the commonest category

[39] Despite its limitations, acknowledged in 10.1.4 above, Success Rate is used here as it is the only quality measure obtained from all of these programs.

in the training sample for every case in the test sample, from the success rate actually observed. These figures tell a different story. MAGS, the only content-based discrimination problem used, is clearly the easiest. Then come all three authorship problems -- each of which has a higher differential score than **any** of the four chronology problems. The worst of the authorship differential scores is more than double the best of the chronology scores. There may be minor objections to this rather simplistic procedure of averaging across results from several different programs or to using the `default' as a true expected value, as done here; but they can hardly be strong enough to overturn such a decisive result.

The relative difficulty of dating a poet's work is in conformity with the results of Martindale (1990), who studied temporal variation in British poetry. While he found plenty of evidence of systematic and predictable variation over a period of five centuries, he also found that "the content of most poets' verse does not change massively across the course of a lifetime".

Although this finding must remain only tentative, since (1) only a single content-based problem (MAGS) was used, and (2) the authorship and chronology problems both involved poetry whereas the content problem involved prose, it is highly suggestive. At the very least it provides a conjecture that deserves further investigation.

10.3 Failings and Future Prospects

Some other lines possibly deserving future investigation arise from a consideration of the main failings of this project.

10.3.1 The Search for Stylograms Continues

Although bringing the previously tacit idea of a stylogram into the foreground is in itself a non-trivial contribution to this field, it must be said that none of the stylistic representation schemes developed as part of this project is entirely satisfactory.

In this respect the efforts made, particularly in Chapters 6 and 8, yielded disappointing results when compared with the work expended. Certainly there were some tantalizing glimpses of what might be achieved by further progress along this road (see, for instance, Figure 6.2 or Table 8.24) but the goal of a `stylogrammar' that combines clarity of interpretation with high classification accuracy remains obstinately out of reach.

As discussed in 10.2.1, the reasons behind this falling short may turn out to be quite profound: there is reason to suspect that the conflict between brevity and effectiveness in text categorization is even more severe than in most other fields where adaptive classification has been tried. This implies that the attempt to marry transparency with accuracy was, if not doomed from the start, then at least more difficult than originally anticipated.

Even this relative failure, however, has its uses. It delineates more clearly than before the challenge facing workers in this field -- to devise a representation scheme that respects the peculiarities of text by accommodating many (but rare) indicators while still being intelligible to human inspection.

GLADRAGS may still prove a suitable vehicle for further progress on this front. The heap-based representation actually used in Chapter 8 is by no means the only one that could be fitted into the GLADRAGS framework. Now that the peculiarities of text categorization are better understood, progress can be expected within this framework using a `flatter' but `wider' representation scheme. Certainly more research in this area is needed.

10.3.2 Towards a Better Benchmark Suite

The idea of a benchmark suite is undoubtedly a good idea which deserves wider acceptance. Its realization here, however, was flawed in several respects -- inevitably so given that it was breaking new ground. Some of these flaws are constructively criticized below (an easy task with hindsight).

A few minor defects could be corrected with ease. For instance, the inclusion of John Jay in the FEDS problem was counter-productive. As indicated by the results of Chapter 9 this merely made an otherwise classic problem harder and less relevant, for uninteresting reasons. In future versions a Federalist problem should be present but without Jay's inclusion. Another easily corrected deficiency is the fact that both AGE2 (with 62 cases) and AGE3 (with 66) were based on samples that were too small: this merely requires collection of more data[40]. In fact, most of the problems (except FEDS) could benefit from slightly larger sample sizes than used here.

The inclusion of a problem involving Snobol4 programs was really only a token gesture towards multi-lingualism. While a generic classifier applicable to many natural languages and to extra-linguistic material as well would obviously be very useful, it is, on reflection, too grandiose a project for the present state of knowledge;

[40] The amount held on disk of Emily Dickinson's verse (AGE2) has been more than doubled in size already, in preparation for a second, improved edition of the benchmark suite.

and the development of a truly representative multi-lingual corpus to test such a thing is a task that would be beyond any individual and probably beyond most institutions. Thus for a second edition of this benchmark suite it would no doubt be more sensible to concentrate on the English language, which presents enough of a challenge already and which is in any case the world's most pervasive medium of international communication.

Even taking the whole range of English as a domain is perhaps over-ambitious, and this is reflected in the rather hotch-potch nature of the present benchmark suite -- containing verse, technical prose, and artificial language, covering, though not without gaps, several types of classification task. This suggests that future benchmark suites should be divided into more coherent sub-suites, with authorship and chronology problems, for instance, no longer lumped together. There is, after all, no reason to expect that the same approach will work on both these problems types; indeed chronology problems might well be more naturally considered within the realm of regression than that of classification. Thus it might even be wise to drop chronology problems from the classification suite, though they could be part of another benchmark suite (possibly, to share a task that could become rather burdensome, hived off to another site). It might also prove sensible to concentrate predominantly on poetry, which already forms the bulk of the present suite, at the expense of prose.

The idea of including some made-up quasi-random data was again a good one, but not particularly well executed. ART1 turned out to be just too easy while ART2 turned out to be rather trivial. With hindsight, it is blindingly obvious that an opportunity was missed here -- the opportunity to include completely patternless data. Thus in future editions there will definitely be synthetic data, but it will be more nearly random (formed, for example, by randomly drawing from an extensive corpus of sentences by many authors). This will provide a null condition that will give any classifiers tested on the suite an opportunity to show that they can detect absence of pattern as well as its presence. Behaviour in the face of pure randomness is likely to be more revealing than that elicited by either ART1 or ART2.

As far as the actual contents of the suite are concerned, the present emphasis on the Forsyth family would be unsuitable in a public-domain resource, and will certainly be reduced when (and if) a public version of this suite is released. In saying this, it perhaps should be noted that if LIT2, the problem involving distinguishing between three related authors, is dropped then some very similar substitute will need to be found, for instance involving work by the three Brontë sisters (plus perhaps their brother Branwell). Indeed there are grounds for arguing that LIT2, extended with slightly more data, might still be worthy of inclusion alongside a Brontë problem, for comparative purposes. So cutting down on the Forsythian bias does not necessarily mean that all Forsyth-related data needs to be thrown away.

This connects with an issue which has not much been discussed: it is my view that in corpus management (which is part of what is involved in putting together a satisfactory text benchmark suite) it is most important for the person in charge of organizing and maintaining the corpus or text-bank to be familiar with its contents. This is only an opinion, but my experience with the Oxford Text Archive and with Project Gutenberg leads me to believe that merely downloading text files over the Internet and dividing them up is asking for trouble. Despite various proposals for standardized mark-up languages, inconsistencies of editing and orthography abound. If the corpus administrator is unfamiliar with the contents of the texts in his or her corpus, it makes the task of ensuring accuracy and consistency formidable indeed. Thus I personally would prefer a reasonably small and somewhat biased benchmark suite to a more extensive but imperfectly standardized collection. After all, the range of text included is always going to be very narrow in comparison with the huge amount that has been written. Most extant large corpora result, inevitably, from multi-person collaborative projects (see, for example, Aijmer & Altenberg, 1991); but Milic (1990) makes essentially this same point -- in favour of the coherence imposed by an individual or small team -- when arguing that it is preferable for the compiler of a text corpus to take on the low-level chores such as proof-reading, verification and even data entry than to delegate them to a hired hand, both to promote consistency and to learn more about its contents.

Finally, while on the subject of content, it is a definite lacuna that Shakespeare is omitted from the present benchmark suite. The bard simply must appear in any future edition[41].

In saying that better benchmarks are needed the value of having a benchmark suite is affirmed. No doubt the improvement of this benchmark suite will occupy the present author for some years to come.

[41] I have now obtained on disk virtually the complete works of Shakespeare (though in a mixture of modernized and archaic spelling) as well as text by some contemporaries like Spenser; but getting this took longer than anticipated and thus it was not ready for inclusion when the present benchmark suite was `frozen' on 11 August 1994.

10.3.3 The Limitations of Fixed-Length Chunking

Another problem faced but not resolved by the present work is the issue, discussed in Chapter 4, of how best to subdivide samples of text into units of analysis. The approach taken here (other than in Chapter 3) was to treat all texts for classification purposes as being made up of chunks of roughly equal length. This partial solution has the virtue of simplicity, but it is also highly artificial. It enabled comparisons to be made between classifiers on more or less equal terms but made the application of those classifiers to practical problems very awkward.

The obvious alternative to fixed-length chunking is to allow a user to insert special code symbols to indicate natural subdivisions within a text file, e.g. between poems in a collection of poetry. This also has disadvantages. A minor disadvantage is that preprocessing by a user is required, which implies extra work and possible mistakes. A more serious drawback is that the task of classifying becomes much harder, because a classifier must deal with considerable disparities in segment length. Since variance is affected by sample size the problem of variability between segments would become more acute.

Nevertheless, in the long term it is probably worth trying to solve this problem and the others brought about by a more natural approach to subdividing text. The limitations of the present approach were exposed in Chapter 9, where none of the systems tested could do directly what a user would clearly want most -- assign categories to individual *Federalist* papers rather than to arbitrary chunks taken from those papers.

10.3.4 The Problem of Cross-Validation

A non-trivial weakness of this project is its failure to incorporate any kind of validatory subsampling -- such as cross-validation or bootstrapping -- into the software developed. Thus subgoal (5) of the original aims (see section 10.1) remains unachieved.

In the benchmarking exercises, a simple division into training and test sets was used, with the same subdivision maintained throughout chapters 4-8 for consistency. This avoids the worst excesses of unrealistic error-rate estimation (as illustrated by the 1-NNC results of Chapter 5) and is adequate for benchmark purposes since any abnormality introduced by one or two exceptional subdivisions will be factored across 13 problems. However more modern methods of unbiased error-rate estimation, based on subsampling, would have used the available data more efficiently. They would also have allowed reliable inferences about individual problems, not just over the 13 problems in aggregate.

This desirable goal was abandoned for two reasons. In the first place, cross-validation and its relatives are very computationally expensive. Apart from BBTC and TEXTREE, which are both fast and deterministic, the learning algorithms employed here were already computer-intensive. It soon became apparent that, without access to a supercomputer, adding cross-validation to them would be impractical. (This constitutes an argument against evolutionary learning systems and another point in favour of BBTC. It also perhaps explains why so many people keep on working with tree-based induction programs despite their demonstrable deficiencies.)

Secondly, the incorporation of cross-validation would have clashed with the choice of equal-length chunks of text as basic units. To give an example: suppose that the leave-1-out method, the simplest form of cross-validation, were being used on a training sample of N cases, i.e. lines. It could well happen that, as individual works are typically split into several lines, training a classifier N times on N subsets of N-1 lines, with the left-over line used each time as an unseen test case, would in fact frequently involve training on most of a single work and then testing on a small part of that same work. This would compromise the main strength of the method -- its ability to deliver relatively unbiased error-rate estimates -- hence rendering the extra work involved rather pointless.

This additional disadvantage of fixed-length chunking was unforeseen. It reinforces the conclusion, stated above in 10.3.3, that division of text into equal-sized chunks will have to be abandoned, despite its convenience, to make further progress.

In addition, further work needs to be done to render GA-based machine learning more compatible with statistical methods of systematic subsampling, such as cross-validation. An obvious approach might be to rely on parallelism, especially since genetic algorithms are quite easy to parallelize. However, despite many fanfares over the years, no single form of parallel computing has ever managed to threaten the worldwide dominance of the Von Neumann architecture. Parallel computers remain confined to niche markets. While creeping parallelism -- allocation of specific functions, e.g. control over disk access or control of video display, to dedicated processors -- continues to make gradual headway, the preeminence of serial processing remains a fact of computing life. For most users computers are serial machines; and this is likely to remain so for the foreseeable future. Thus a solution based on parallel hardware will not bring genetic adaptive text-categorization into the mainstream.

If a software solution is to be sought, entropy optimization, in the form of the Minimum-Message-Length (MML) principle (mentioned at various points throughout this thesis), appears to offer the best hope. Many researchers have worked on aspects of entropy optimization (e.g. Bichsel & Seitz, 1989; Wolff, 1991; Kapur & Kesavan, 1992; Quinlan, 1993) including Forsyth et al. (1994). In the current context, the important point about the MML principle is that it points to the possibility of achieving the same effect as cross-validation without doing the extra resampling work. A small-scale example is the entropy-based pruning of TEXTREE reported in 6.2.2. This achieved essentially what the designers of the CART software (Breiman et al., 1984) achieve through cross-validation, but at a fraction of the computational cost. Further progress in this area, e.g. in developing theoretically rigorous MML-based fitness functions, could render cross-validation obsolete. This would make evolutionary machine learning a more attractive proposition, and would be desirable on other grounds as well.

10.3.5 Other Potential Improvements

This work has also exhibited several other minor deficiencies, as is to be expected, some of which are reviewed below.

Iota (the Index of textual association) proved quite effective (see section 8.5) but it rests on shaky theoretical foundations. Although the F-ratio and the Chi-squared statistics are mathematically related, their combination in a single measure was made here on pragmatic grounds, and has no rigorous justification. It could turn out on deeper analysis to be a mathematical solecism. More work is needed in this area, to find better indices of this type and to explain and justify their success. A start could possibly be made by investigating the consquences of remaining consistently within the Analysis-of-Variance paradigm rather than using a hybrid measure. In this framework, the problem of combining distinctiveness with consistency would be formulated in terms of two factors: T (the text types involved) and P (the position within the data sequence). If split-half testing were being used then P would have only two levels, though in general it could have three or four if enough data were present. A more uniform formulation of Iota (still employing omega-squared as a measure of strength of association based on variance) would then be

$$\omega_T^2 - \omega_P^2 - \omega_{TP}^2$$

where a measure of the main effect attributable to P and the interaction effect of text type with position are both deducted from the main effect of text type. Of course, this usage still falls outside the classical ANOVA model, based on hypothesis testing, but at least commensurable quantities are being related.

Other approaches, e.g. using conditional entropy, are also conceivable. Certainly there is plenty of scope here for more work and, with an index that is already quite promising, the payoff from further improvement would be considerable.

On a related theme, much more work is needed on quality measures like Brier score, logarithmic penalty and suchlike. The introduction of such measures is a useful first step, but more needs to be done, especially on baselining. For instance, the expected Brier Score of a purely random classifier depends on the number of categories involved. This has not been compensated for in the present work nor in previous research using Brier Scoring to reward human forecasters (where comparison between tasks with unequal number of categories is not normally necessary).

Another area for further research is text classification by cross-codebook compression. In Chapter 3 this method proved disappointing. Though it prompted the development of Monte-Carlo feature-finding with short substrings, which was a success, it did not prove very effective in itself. However it is such a neat idea that it deserves another look, perhaps using other data-compression algorithms.

IOGA, which (as mentioned in 10.1.4) is one of the positive contributions of this project, was rather slow in execution, even in comparison with the other GA-based systems MAWS and GLADRAGS. To be practicable in an applied context, it would need to be speeded up considerably, preferably without resorting to parallel hardware which (as argued above) would confine it to the sidelines. Since Manhattan distance is an additive combination of separate components (and even Euclidean distance is additive apart from the final square-root operation) it might help to precompute and store submatrices giving distances between cases on each variable, although this would be very demanding of memory with high-dimensional data sets. An alternative approach would be to represent archetypes explicitly which, provided that the number of cases and the number of features used in the archetypes were few compared to the total number of cases and features (as would usually be the case), should lead to a decrease in execution time. The representational revision required to effect this change could also allow the algorithm to be modified to admit archetypes that do not occur in the data -- an issue whose pros and cons also deserve fuller investigation.

Finally, on a personal note, the choice of C as the main programming language of this project seems in retrospect a mistake. It is simply unsuited to the kind of rapid prototyping intrinsic to research-driven software development. By contrast, using Spitbol (a dialect of Snobol4), for which a translator was not available until this project was well advanced, felt like having a `secret weapon', even though it is a DOS-based system not incorporating the latest ideas on human interfacing or on software-project management. However Spitbol produces vast .EXE files, as it includes most of the interpreter within a compiled module, and so is best used in interpretive mode during development rather than as a compiler for production versions. In future a compromise language like Visual Basic, which is less user-hostile than C but also produces fairly efficient object code, might be a better choice.

10.4 Closing Remarks

10.4.1 How Desirable Were the Desiderata?

Certain desiderata which form a kind of backdrop to this work were accepted from the outset. These may be seen as boundary conditions or even biases. I prefer to view them as self-imposed limitations accepted to keep the task manageable. They have been mentioned in earlier chapters but have not been gathered together, so they are listed explicitly here for reference.

Desiderata

(1) More attention than usual should be paid to the **sequential** nature of text.

(2) More attention than usual should be paid to the **representation** of systematic differences between text types.

(3) Systems developed should be **generic**: they should not be restricted to the English language.

(4) Text should in general be treated as text, with a **minimum of pre-processing**.

(5) There should be a concentration on methods of machine learning that have been relatively neglected by those attempts that have so far been made to bring machine-learning techniques into stylometry. (In practice this means: **avoidance of connectionist systems generally and back-propagation especially**.)

This last item can be justified on the principle that research should be exploratory rather than exploitative. The others could perhaps also be justified by various forms of argument, but the main question here is: how fruitful were they?

Desideratum (1) was more of a hindrance than a help. The primacy of digrams over digraphs, as well as the failure, in Chapter 4, of the SEQ1 and SEQ2 programs -- which worked with word-transitions -- to attain the same level of performance as BBTC using characters, suggests that, as far as categorization is concerned, the subtle pattern of sequential relationships among distant textual elements so beloved of Chomskyan linguists is of little practical importance. Although collocations occasionally proved diagnostic (and are not excluded from the Monte-Carlo feature-finding process employed here), much of the information needed to detect stylistic peculiarities can be found in surprisingly short sequences, often shorter than a single word. The development of more robust parsers may eventually force a reappraisal of this conclusion, but until such things are more reliable and more widespread I intend to continue working with short substrings. My advice to other researchers would be to do likewise.

Desideratum (2) was essential to this project: it would have been a totally different project without this emphasis on representation. However, it paid off less handsomely than expected. With hindsight it can be seen that the reason that this methodological preference was less fruitful than it should have been was a perseverance with formalisms sanctioned by previous machine-learning practice, such as trees and hierarchical logical/arithmetic formulae, after the evidence (e.g. from the performance of BBTC and MAWS) was already starting to indicate that different (less complex) formalisms are required in this area. So this emphasis should not be abandoned. Indeed there is a case to be made that it should be strengthened, and that a deeper investigation of the forms of representation needed for effective textual categorization should become a research priority.

There are arguments and counter-arguments concerning desideratum (3). As an ideal the notion of a generic text classifier, not tied to any particular language, served a motivational purpose in this research, e.g. in motivating the development of Monte-Carlo feature-finding which, as argued in 10.1.1, is both novel and useful. But, while there will always be a place for generic classifiers, in any specific application they are likely to ignore potentially useful information. So while there is a need for further development of generic text classifiers, such as BBTC, MAWS and GLADRAGS, this should not be thought to preclude research into more specialized systems. A classifier that could only handle English text would still have a wide range of applicability, and it could make use of syntactic and semantic information as well as lexical. It is possible to envisage a system that could construct classification rules incorporating whatever combination of

- function-word frequencies (cf. Mosteller & Wallace, 1984);
- short substrings, including collocations (as in this thesis);
- vocabulary richness measures (cf. Holmes, 1992);
- semantic category counts (cf. McKenzie & Martindale, 1995);
- syntactic category counts (cf. Ross & Hunter, 1994);

proved most effective. Such a system would potentially offer extremely high-quality text categorization -- especially if transition frequencies between syntactic and/or semantic categories were also used. Of course the difficulties of coordinating several information sources would be great, but overcoming these difficulties would be a worthwhile research objective. Until quite recently the software to provide such information (particularly syntactic and semantic) from unrestricted real-world texts was neither reliable nor widely available; but that situation is beginning to change. This suggests that, if any of the five preconditions discussed in this section are to be dropped, dropping this one will probably have the most liberating effect.

Desideratum (4), a preference for minimal pre-processing, is hard to evaluate. At least it proved feasible to classify lines of ASCII text without first turning them into numeric feature vectors. The main problem with the rather well-trodden route of counting words in text and then turning those word counts into vectors of numeric features to be processed by conventional statistical methods is a practical one: such vectors tend to be very long (unless someone has already applied a good deal of domain knowledge to reduce the number of variables). For example, using digram frequencies would simply exceed the limits of many current multi-variate software packages. And if digram frequencies did not bust the software, trigram or tetragram frequencies surely would. On the other hand, sticking rigidly to text as text cuts an investigator off from many standard techniques of data analysis and presentation which would be helpful in any practical application. Thus, though on balance desideratum (4) was probably useful in this study, there is no need to stick to it slavishly. In fact, further research into ways of integrating the pure-text and the vector-based approaches would be most valuable.

Desideratum (5) proved its worth heuristically. At present there is a general surfeit of work on artificial neural nets, and in stylometry neural networks are the only kind of learning systems to have been tried. So there is a need for some workers to investigate other methods of adaptive text classification, as has been done here[42]. Also, neural nets are famously inscrutable, so a connectionist approach would conflict to some extent with desideratum (2). These are contingent facts, but until they change this last desideratum will serve a useful purpose. It is thus probably worth carrying over to future work.

So to build on the base provided by the present work, desideratum (1) should be scrapped while (3) may be relaxed somewhat as suitable supporting software becomes available. The other three should be retained, at least in the medium term.

[42] Although BBTC could no doubt (to a receptive audience) be presented as some sort of 1-layer `probabilistic neural net', such a portrayal would be disingenuous.

10.4.2 Last Words on Words

Although, as emphasized in Chapter 1, there is no shortage of text in this world, history may very well show that the Golden Age of text is already over, or at least has passed its peak. If the projections of the cyber-surfers are to be believed, the time will soon come when we no longer read books, newspapers or magazines -- not on paper anyway. What we will want to know is where to find information on the Internet; and when we find it, it will be mainly in forms other than text, such as digitized images and sound.

Even with written material the text version will no longer be canonical: it will merely be the by-product of interpreting a program in a page-description language like PostScript. The `master copy' (in so far as such a thing exists) will be a long list of instructions, only a small proportion of which will consist of grammatical sentences in a natural language.

In such a context it might be thought that the need for automatic text categorization will fade away, and thus that the present work is no more than a nostalgic backward glance at a lost era of literacy. However, prognostications are notoriously fallible and even if text as such is destined for a lesser role than formerly in the multi-mediumistic future, there will always remain over five centuries' worth of print-based cultural artefacts to be analyzed -- if only out of antiquarian interest. Thus a need for text processing will endure. In addition, even multi/hyper-media `documents' will no doubt still contain some textual information, leading to a requirement for text analysis to be linked with image-processing and other signal-processing techniques within a more general analytic framework.

So the need for text analysis will not disappear rapidly, if it ever does. Thus developing better text-categorization software will remain useful for some time to come. The machine-learning community will undoubtedly benefit if it takes up this challenge and pays more attention to this important field, for one of the lessons of this project is that adaptive text-categorization poses interesting and difficult problems that have only been tackled tangentially in previous work on pattern recognition using primarily numeric data. Solving such problems will enrich the discipline.

I began this thesis thinking that stylometers had much to learn about machine learning. I now think that the machine-learning community has more to learn from stylometry than vice versa. I hope that this research will prove to have played a part in furthering both of these ends.

Reference List

Name-Date citations have been used throughout the foregoing text. The style adopted in the reference list below is as recommended in "Instructions to Authors" of the journal *Pattern Recognition Letters*, published by North-Holland.

Ackley, D.H. (1987). An Empirical Study of Bit Vector Function Optimization. In: L. Davis, ed., *Genetic Algorithms & Simulated Annealing*. Pitman, London.

Adamson, G.W. & Boreham, J. (1974). The Use of an Association Measure Based on Character Structure to Identify Semantically Related Pairs of Words and Document Titles. *Information Storage & Retrieval*, 10, 253-260.

Afifi, A.A. & Azen, S.P. (1979). *Statistical Analysis: a Computer Oriented Approach*, 2nd. edition, Academic Press, New York.

Aha, D.W., Kibler, D. & Albert, M.K. (1991). Instance-Based Learning Algorithms. *Machine Learning*, 6, 37-66.

Ahonen, H., Mannila, H. & Nikunen, E. (1993). Forming Grammars for Structured Documents. In: *Proc. AAAI-93 Workshop on Knowledge Discovery in Databases*. AAAI Press, Menlo Park, CA.

Aijmer, K. & Altenberg, B. (1991) eds. *English Corpus Linguistics*. Longman, London.

Aleksander, I. & Morton, H. (1990). *An Introduction to Neural Computing*. Chapman & Hall, London.

Aleksander, I. & Stonham, T.J. (1979). A Guide to Pattern Recognition using Random-Access Memories. *IEEE J. Computers & Digital Techniques*, 2(1), 120-124.

Allaway, S.L., Ritchie, C.D., Robinson, D. & Smolski, O.R. (1988). Detection of Alcohol-Induced Fatty Liver by Computerized Tomography. *J. Royal Soc. Medicine*, 81, 149-151.

Allinson, N.M. (1994). Personal Communication. [from: Image Engineering Laboratory, University of York.]

Althoff, K-D., Auriol, E., Barletta, R. & Manago, M. (1995). *A Review of Industrial Case-Based Reasoning Tools*. AI Intelligence, Oxford.

Anderson, E. (1935). The Irises of the Gaspe Peninsula. *Bulletin of the American Iris Society*, 59, 2-5.

Andrews, D.F. & Herzberg, A.M. (1985). *Data: a Collection of Problems from Many Fields for the Student and Research Worker*. Springer-Verlag, New York.

Apté, C., Damerau, F. & Weiss, S.M. (1993). Knowledge Discovery for Document Classification. In: *Proc. AAAI-93 Workshop on Knowledge Discovery in Databases*. AAAI Press, Menlo Park, CA.

Arps, R.B. (1979). Binary Image Compression. In: W.K. Pratt, ed. *Image Transmission Techniques*. Academic Press, New York.

Attneave, F. (1959). *Applications of Information Theory to Psychology*. Holt, New York.

Axelrod, R. (1984). *The Evolution of Cooperation*. Basic Books, New York.

Axelrod, R. & Hamilton, W.D. (1981). *The Evolution of Cooperation*. Science, 211, 1390-1396.

Back, T., Hoffmeister, F. & Schwefel, H.P. (1991). A Survey of Evolution Strategies. *Proc. Fourth Internat. Conf. on Genetic Algorithms & their Applications*. Morgan Kaufmann Publishers, 2-9.

Baillie, W.M. (1974). Authorship Attribution in Jacobean Dramatic Texts. In: J.L. Mitchell, ed., *Computers in the Humanities*, Edinburgh Univ. Press.

Batchelor, B.G. (1974). *Practical Approaches to Pattern Classification*. Plenum Press, London.

Batchelor, B.G. (1978) ed. *Pattern Recognition: Ideas in Practice*. Plenum Press, New York.

Bayes, T. (1763). An Essay towards Solving a Problem in the Doctrine of Chances. *Philosophical Trans. Royal Society*, 53, 370-418. [Reprinted in: E.S. Pearson & M.G. Kendall (1970) eds. *Studies in the History of Statistics & Probability*. Griffin, London.]

Beale, R. & Jackson, T. (1990). *Neural Computing: an Introduction*. Adam Hilger, Bristol.

Bechhofer, R.E. & Dunnett, C.W. (1988). *Percentage Points of Multivariate Student t Distributions*. Selected Tables in Mathematical Studies, Volume 11: American Mathematical Society, Providence, R.I.

Berwick, R.C. & Pilato, S. (1987). Learning Syntax by Automata Induction. *Machine Learning*, 2, 9-38.

Bichsel, M. & Seitz, P. (1989). Minimum Class Entropy: a Maximum Information Approach to Layered Networks. *Neural Networks*, 2, 133-141.

Binongo, J.N.G. (1994). Joaquin's Joaquinesquerie, Joaquinesquerie's Joaquin: A Statistical Expression of a Filipino Writer's Style. *Literary & Linguistic Computing*, 9(4), 267-279.

Bledsoe, W.W. & Browning, I. (1959). Pattern Recognition and Reading by Machine. *Proc. Eastern Joint Computer Conf.*, Boston, Mass., 225-232. [Reprinted in: L. Uhr (1966) ed., *Pattern Recognition*. John Wiley & Sons, New York.]

Brainerd, B. (1979). Pronouns and Genre in Shakespeare's Drama. *Computers & the Humanities*, 13, 3-16.

Brainerd, B. (1980). The Chronology of Shakespeare's Plays: a Statistical Study. *Computers & the Humanities*, 14, 221-230.

Bratko, I. & Kononenko, I. (1986). Learning Diagnostic Rules from Incomplete & Noisy Data. *AI Methods in Statistics*. Unicom Seminars, London, December 1986.

Breiman, L., Friedman, J.H., Olshen, R.A. & Stone, C.J. (1984). *Classification and Regression Trees*. Wadsworth, Monterey, California.

Brier, G.W. (1950). Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, 78, 1-3.

Brooks, F.P. (1982). *The Mythical Man-Month: Essays on Software Engineering,* 2nd. edition. Addison-Wesley, Reading Massachusetts.

Brunet, E. (1978). *Vocabulaire de Jean Giraudoux: Structure et Evolution*. Slatkine, Geneva.

Buntine, W. (1992). Learning Classification Trees. *Statistics & Computing*, 2, 63-73.

Burrows, J.F. (1992). Not unless you Ask Nicely: the Interpretive Nexus between Analysis and Information. *Literary & Linguistic Computing*, 7(2), 91-109.

Burrows, J.F. & Craig, D.H. (1994). Lyrical Drama and the "Turbid Montebanks": Styles of Dialogue in Romantic and Renaissance Tragedy. *Computers & the Humanities*, 28, 63-86.

Butler, C.S. (1979). Poetry and the Computer: some Quantitative Aspects of the Style of Sylvia Plath. *Proc. British Academy*, LXV, 291-312.

Cestnik, B., Kononenko, I. & Bratko, I. (1987). ASSISTANT 86: A Knowledge-Elicitation Tool for Sophisticated Users. In: I. Bratko & N. Lavrac, eds., *Progress in Machine Learning*. Sigma Press, Wilmslow, Cheshire.

Chang, C.L. (1974). Finding Prototypes for Nearest Neighbour Classifiers. *IEEE Trans. on Computers*, C-23(11), 1179-1184.

Cheeseman, P. (1990). On Finding the Most Probable Model. In: J. Shrager & P. Langley, eds., *Computational Methods of Scientific Discovery and Theory Formation*. Morgan Kaufmann, San Mateo, California.

Chou, P.A. (1991). Optimal Partitioning for Classification and Regression Trees. IEEE *Trans. on Pattern Analysis & Machine Intelligence*, PAMI-13(4), 340-354.

Ciampi, A., du Berger, R., Taylor, H.G. & Thiffault, J. (1991). RECPAM: a Computer Program for Recursive Partition and Amalgamation for Survival Data and Other Situations Frequently Occurring in Biostatistics. *Comp. Methods & Programs in Biomedicine*, 36, 51-61.

Clark, P. & Niblett, T. (1987). Induction in Noisy Domains. In: I. Bratko & N. Lavrac, eds., *Progress in Machine Learning*. Sigma Press, Wilmslow.

Clarke, D.D. (1995). Personal Communication. [from: Department of Psychology, University of Nottingham.]

Cooke, R.M. (1991). *Experts in Uncertainty*. Oxford University Press, Oxford.

Cost, S. & Salzberg, S. (1993). A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning*, 10, 57-78.

Crawford, S.L. (1989). Extensions to the CART Algorithm. *Int. J. Man-Machine Studies*, 31, 197-217.

Cupit, J. & Shadbolt, N.R. (1994). Representational Redescription within Knowledge Intensive Data-Mining. In: R. Mizoguchi, H. Motoda, J. Boose, B. Gaines & R. Compton, eds., *Proc. Japanese Knowledge Acquisition Workshop*, JKAW-94, Osaka University, 121-135.

D* (1974). *Stemya "Tichogo Dona" (Zagadki romana)*. YMCA-Press, Paris.

Darwin, C.R. & Wallace, A.R. (1858). On the Tendency of Species to Form Varieties; and on the Perpetuation of Varieties and Species by Natural Means of Selection. *Paper* *presented to the London Linnean Society, 1st July 1858*. In: D.C. Porter & P.W. Graham (1993). *The Portable Darwin*. Penguin Books, London, 86-104.

Dasarathy, B.V. (1991) ed. *Nearest Neighbour (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California.

Davis, L. (1991) ed. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.

Dawkins, R. (1976). Hierarchical Organisation: a Candidate Principle for Ethology. In: P.P.G. Bateson & R.A. Hinde, eds., *Growing Points in Ethology*. Cambridge University Press.

Devijer, P.A. & Kittler, J. (1982). *Pattern Recognition: a Statistical Approach*. Prentice-Hall, New Jersey.

Dylan, B. (1988). *Knocked Out Loaded*. Sony Music Entertainment Inc.

Dylan, B. (1989). *Oh Mercy*. CBS Records Inc.

Dylan, B. (1994). *Lyrics 1962-1985*. Harper Collins Publishers, London. [original U.S. edition published 1985.]

Edwards, D. (1992). Personal Communication. [from: Department of Mathematical Sciences, UWE, Bristol.]

Edwards, E. (1964). *Information Transmission*. Chapman & Hall, London.

Efron, B. (1983). Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation. *J. American Statist. Assoc.*, 78, 316-330.

Efron, B. & Gong, G. (1983). A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation. *The American Statistician*, 37(1), 36-48.

Efron, B. & Thisted, R. (1976). Estimating the Number of Unseen Species: How many Words did Shakespeare Know? *Biometrika*, 63, 435-437.

Eliot, T.S. (1963). *Collected Poems 1909-1962*. Faber & Faber Limited, London.

Eliot, V. (1971) ed. *The Waste Land, Facsimilie and Transcript*. Faber & Faber, London.

Ellegaard, A. (1962). *Who was Junius?* Almqvist & Wiksell, Stockholm.

Elliott, W.E.Y. & Valenza, R.J. (1991). A Touchstone for the Bard. *Computers & the*

Humanities, 25, 199-209.

Elliott, W.E.Y. & Valenza, R.J. (1995). *And Then there Were None: Winnowing the Shakespeare Claimants*. Claremont McKenna College, Claremont CA.

Emmer, M.B., Quillen, E.K. & Dewar, R.B.K. (1989). *Macro Spitbol: Tutorial and Program Reference Manual*. Catspaw, Inc., Salida, Colorado.

Fang, A.C. & Nelson, G. (1994). Tagging the Survey Corpus: a LOB to ICE Experiment using AUTASYS. *Literary & Linguistic Computing*, 9(3), 1994.

Feynman, R.P. & Leighton, R. (1992). *"Surely You're Joking Mr. Feynman": Adventures of a Curious Character*. Vintage Books, London.

Fisher, R.A. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7, 179-188.

Fisher, R.A. (1938). The Statistical Utilization of Multiple Measurements. *Annals of Eugenics*, 8, 376-386.

Fix, E. & Hodges, J.L. (1951). Discriminatory Analysis -- Nonparametric Discrimination: Consistency Properties. *Project 21-49-004, Report No. 4*, USAF School of Aviation Medicine, Randolf Field, Texas, 261-279.

Flury, B. & Riedwyl, H. (1988). *Multivariate Statistics: a Practical Approach*. Chapman & Hall, London.

Fogarty, T.C. (1992). First Nearest Neighbor Classification on Frey & Slate's Letter Recognition Problem. *Machine Learning*, 9, 387-388.

Fogarty, T.C. & Ireson, N.S. (1994). Evolving Bayesian Classifiers for Credit Control -- a Comparison with other Machine-Learning Methods. *IMA J. of Mathematics Applied in Business & Industry*, 5, 63-75.

Fogel, D.B. (1993). Applying Evolutionary Programming to Selected Traveling Salesman Problems. *Cybernetics & Systems*, 24, 27-36.

Forsyth, E.H. (1994). *Enamelled in Fire: Collected Poems*. Sutton Print Partners, Nottingham.

Forsyth, J.L. (1989). *On Such a Day as This...*. Grainloft Books, Sussex.

Forsyth, J.L. (1990). *From Time to Time*. Grainloft Books, Sussex.

Forsyth, R.S. (1978). *The BASIC Idea*. Chapman & Hall Ltd., London.

Forsyth, R.S. (1981). *BEAGLE -- a Darwinian Approach to Pattern Recognition*. Kybernetes, 10, 159-166.

Forsyth, R.S. (1989) ed. *Machine Learning: Principles & Techniques*. Chapman & Hall, London.

Forsyth, R.S. (1990). Neural Learning Algorithms: Some Empirical Trials. *Proc. 3rd International Conf. on Neural Networks & their Applications, Neuro-Nimes-90*. EC2, Nanterre.

Forsyth, R.S. (1992). Ockham's Razor as a Gardening Tool: Simplifying Discrimination Trees by Entropy Minimax: In: M.A. Bramer & R.W. Milne, eds., *Research & Development in Expert Systems IX*. Cambridge University Press, Cambridge.

Forsyth, R.S. (1995). *Stylistic Structures: a Computational Approach to Text Classification*. Unpublished Doctoral Thesis, Faculty of Science, University of Nottingham. [Self-Reference.]

Forsyth, R.S., Clarke, D.D. & Wright, R.L. (1994). Overfitting Revisited: an Information Theoretic Approach to Simplifying Discrimination Trees. *J. Experimental & Theoretical Artificial Intelligence*, 6, 289-302.

Forsyth, R.S. & Rada, R. (1986). *Machine Learning: Applications in Expert Systems and Information Retrieval*. Ellis Horwood, Chichester.

Fries, C.C. (1952). *The Structure of English*. Harcourt-Brace, N.Y.

Fukunaga, K. & Mantock, J.M. (1984). Nonparametric Data Reduction. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, PAMI-6(1), 115-118.

Gabor, G. (1975). The eta-NN Method: a Sequential Feature Selection for Nearest Neighbour Decision Rule. In: I. Csiszar & P. Elias, eds., *Topics in Information Theory*. North-Holland, Amsterdam.

Gammerman, A. & Thatcher, A.R. (1991). Bayesian Diagnostic Probabilities without Assuming Independence of Symptoms. *Methods of Information In Medicine*, 30(1), 15-22.

Gerlenter, D. (1994). *The Muse in the Machine*. Fourth Estate, London.

Geva, S. & Sitte, J. (1991). Adaptive Nearest Neighbor Pattern Classification. *IEEE Trans. on Neural Networks*, NN-2(2), 318-322.

Gimpel, J.F. (1986). *Algorithms in Snobol4*. Catspaw Inc., Salida, Colorado. [first edition: John Wiley & Sons / Bell Telephone Labs, 1976.]

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading, Mass.

Goldberg, J.L. (1995). *A Learning Method for Text Categorization*. Technical Report, Dept. Computer Science, Texas A & M University, College Station, Texas.

Goodman, M. (1990). Prism: a Case-Based Telex Classifier. In: A. Rappaport & R. Smith, eds., *Innovative Applications of Artificial Intelligence*, volume. 2. MIT Press, Cambridge, Mass.

Goodwin, P. & Wright, G. (1991). *Decision Analysis for Management Judgment*. Wiley, Chichester.

Gottman, J.M. & Roy, A.K. (1990). *Sequential Analysis*. Cambridge University Press, Cambridge.

Gowers, J.I. (1995). Personal Communication. [from: Dept. of Mathematical Sciences, University of the West of England.]

Grishman, R. (1986). *Computational Linguistics*. Cambridge University Press, Cambridge.

Griswold, R.E. (1975). *String and List Processing in Snobol4*. Prentice-Hall Inc., Englewood Cliffs, N.J.

Griswold, R.E., Poage, J.F. & Polonsky, I.P. (1971). *The Snobol4 Programming Language*, 2nd. edition: Prentice-Hall, Englewood Cliffs, N.J.

Hamilton, A., Madison, J. & Jay, J. (1961). *The Federalist Papers*. Mentor Books, New York. [edition of Clinton Rossiter; first edition, 1788.]

Hand, D.J. (1994). In: Discussion of a Paper Presented by B.D. Ripley. *J. Royal Statist. Society (B)*, 56(3), 437-456.

Hand, D.J. & Batchelor, B.G. (1978). An Edited Nearest Neighbour Rule. *Information Sciences*, 14, 171-180.

Hart, A.E. (1985). Experience in the Use of an Inductive System in Knowledge Engineering. In: M.A. Bramer, ed., *Research & Development in Expert Systems*. Cambridge University Press, 117-125.

Hart, P.E. (1968). The Condensed Nearest Neighbour Rule. *IEEE Trans. on Info. Theory*, IT-14(3), 515-516.

Hayes, P. & Weinstein, S. (1991). Adding Value to Financial News by Computer. In: *Proc. First Internat. Conf. on Artificial Intelligence on Wall Street*, 2-8.

Hays, W.L. (1969). *Statistics*. Holt, Rinehart & Winston, International Edition, London.

Hayter, A.J. (1984). A Proof of the Conjecture that the Tukey-Kramer Multiple Comparisons Procedure is Conservative. *Annals of Statistics*, 12, 61-75.

Heikkila, J. & Voutilainen, A. (1993). ENGCG: an Efficient and Accurate Parser for English Texts. In: *Proc. 1993 Joint Internat. Conf. of Assoc. for CATH & ALLC* at Georgetown Univ., Washington D.C., 67-70.

Held, G. & Marshall, T.R. (1991). *Data Compression*, 3rd edition. Wiley, Chichester.

Herdan, G. (1965). Comment on the Paper by Mr Morton. *J. Royal Statistical Society (A)*, 128(2), 229-231.

Hilton, M.L. & Holmes, D.I. (1993). An Assessment of Cumulative Sum Charts for Authorship Attribution. *Literary & Linguistic Computing*, 8(2), 73-80.

Hintzman, D.L. (1986). "Schema Abstraction" in a Multiple-Trace Memory Model. *Psychological Review*, 93(4), 411-428.

Hockey, S. (1980). *A Guide to Computer Applications in the Humanities*. Duckworth, London.

Hogg, R.V. & Ledolter, J. (1992). *Applied Statistics for Engineers & Physical Scientists*. Macmillan, New York.

Holland, J.H. (1975). *Adaptation in Natural & Artficial Systems*. Univ. Michigan Press, Ann Arbor.

Holland, J.H., Holyoak, K.J., Nisbett, R.E. & Thagard, P.R. (1986). *Induction: Processes of Inference, Learning and Discovery*. MIT Press, Cambridge, Mass.

Holmes, D.I. (1985). The Analysis of Literary Style -- a Review. *J. Royal Statistical Society (A)*, 148(4), 328-341.

Holmes, D.I. (1992). A Stylometric Analysis of Mormon Scripture and Related Texts. *J. Royal Statistical Society (A)*, 155(1), 91-120.

Holmes, D.I. (1994). Authorship Attribution. *Computers & the Humanities*, 28, 1-20.

Holmes, D.I. & Forsyth, R.S. (1995). The 'Federalist' Revisited: New Directions in Authorship Attribution. *Literary & Linguistic Computing*, 10(2), 111-127.

Holte, R.C. (1993). Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning*, 11, 63-91.

Honoré, A. (1979). Some Simple Measures of Richness of Vocabulary. *ALLC Bulletin*, 7(2), 172-177.

Horton, T.B. (1987). Doctoral Thesis, University of Edinburgh.

Hughes, E.J. (1993). *A Choice of Emily Dickinson's Verse*. Faber & Faber Limited, London.

Hunt, E.B., Marin, J. & Stone, P.I. (1966). *Experiments in Induction*. Academic Press, New York.

Hutchinson, J. & Palliser, D.M. (1980). *Bartholomew City Guides: York*. Bartholomew, London.

Indurkhya, N. & Weiss, S.M. (1991). Iterative Rule Induction Methods. *J. Applied Intelligence*, 1, 43-54.

Iversen, G.R. (1984). *Bayesian Statistical Inference*. Sage Publications, Newbury Park, California.

Jacobs, P.S. (1993). Using Statistical Methods to Improve Knowledge-Based News Categorization. *IEEE Expert*, April 1993, 13-23.

James, M. (1985). *Classification Algorithms*. Collins, London.

Jennison, C. & Sheehan, N. (1993). *Theoretical and Empirical Properties of the Genetic Algorithm*. Statistics Research Report 93:06, School of Math. Sciences, University of Bath, BA2 7AY.

Johnson, T.H. (1970) ed. *Emily Dickinson: Collected Poems*. Faber & Faber Limited, London.

Jones, D.S. (1979). *Elementary Information Theory*. Clarendon Press, Oxford.

Jones, S. (1994). *The Language of the Genes*. Harper Collins, London.

Kapur, J.N. & Kesavan, H.K. (1992). *Entropy Optimization Principles with Applications*. Academic Press, San Diego, CA.

Keane, A.J. (1995). *A Brief Comparison of Some Evolutionary Optimization Methods*. Technical Report pl-3.0, Department of Engineering Science, University of Oxford, Parks Road, Oxford OX1 3PJ.

Kelly, J.D. & Davis, L. (1991). Hybridizing the Genetic Algorithm and the K Nearest Neighbors Classification Algorithm. In: R.K. Belew & L.B. Booker, eds., *Proc. Fourth International Conf. on Genetic Algorithms*. Morgan-Kaufmann, San Mateo, California, 377-383.

Kinnear, K.E. (1994) ed. *Advances in Genetic Programming*. MIT Press, Cambridge, Mass.

Kjell, B. (1994). Authorship Determination Using Letter Pair Frequency Features with Neural Net Classifiers. *Literary & Linguistic Computing*, 9(2), 119-124.

Kjetsaa, G. (1979). "And Quiet Flows the Don" through the Computer. *ALLC Bulletin*, 7, 248-256.

Koester, P. (1971). Computer Stylistics: Swift and some Contemporaries. In: R.A. Wisbey, ed., *The Computer in Literary & Linguistic Research*. Cambridge University Press, Cambridge.

Kohonen, T. (1988). *Self-Organization & Associative Memory*, 2nd. edition. Springer-Verlag, Berlin.

Kolodner, J.L. (1993). *Case-Based Reasoning*. Morgan Kaufmann, California.

Koza, J.R. (1992). *Genetic Programming*. MIT Press, Cambridge, Mass.

Langley, P. (1993). Induction of Recursive Bayesian Classifiers. In: P.B. Brazdil, ed., *Machine Learning: ECML-93*. Springer-Verlag, Berlin.

Larsen, W., Rencher, A. & Layton, T. (1980). Who Wrote The Book of Mormon?: an Analsysi of Wordprints. *Brigham Young University Studies*, 20, 225-251.

Lebart, L. & Salem, A. (1994). *Statistique Textuelle*. Dunod, Paris.

Lebart, L., Salem, A. & Berry, L. (1991). Recent Developments in the Statistical Processing of Textual Data. *Applied Stochastic Models & Data Analysis*, 7, 47-62.

Ledger, G.R. (1989). *Re-Counting Plato*. Oxford University Press, Oxford.

Ledger, G.R. & Merriam, T.V.N. (1994). Shakespeare, Fletcher, and the Two Noble Kinsmen. *Literary & Linguistic Computing*, 9(3), 235-248.

Leech, G.N. (1987). General Introduction. In: R. Garside, G. Leech & G. Sampson, eds., *The Computational Analysis of English*. Longman, Harlow, Essex.

Lehnert, W., Soderland, S., Aronow, D., Feng, F. & Shmueli, A. (1995). Inductive Text Classification for Medical Applications. *J. Experimental & Theoretical Artificial Intelligence*, 7(1), 49-80.

Lichtenstein, S., Fischhoff, B. & Phillips, L.D. (1982). Calibration of Probabilities: the State of the Art to 1980. In: D. Kahneman, P. Slovic, & A. Tversky, eds., *Judgment under Uncertainty: Heuristics and Biases*. Cambridge University Press.

Lord, R.D. (1958). De Morgan and the Statistical Study of Literary Style. *Biometrika*, 45, 282.

Lunn, A.D. & McNeil, D.R. (1991). *Computer-Interactive Data Analysis*. John Wiley & Sons, Chichester.

Manly, B.F.J. (1994). *Multivariate Statistical Methods: a Primer*. Chapman & Hall, London.

Mansell, D. (1974). "The Old Man and the Sea" and the Computer. *Computers & the Humanities*, 8, 195-205.

Martindale, C. (1990). *The Clockwork Muse: the Predictability of Artistic Change*. Basic Books, New York.

Martindale, C. & McKenzie, D.P. (1995). On the Utility of Content Analysis in Authorship Attribution: the Federalist. *Computers & the Humanities*, 29, in press.

Masand B., Linoff, G. & Waltz, D. (1992). Classifying News Stories using Memory-Based Reasoning. In: *Proc. 15th Annual Internat. ACM SIGIR Conf. on R & D in Information Retrieval*, June 1992, 59-65.

Matthews, R.A.J. & Merriam, T.V.N. (1993). Neural Computation in Stylometry I: an Application to the Works of Shakespeare and Fletcher. *Literary & Linguistic Computing*, 8(4), 203-209.

Maud, R. (1989) ed. *Dylan Thomas: the Notebook Poems 1930-1934*. J.M. Dent & Sons Limited, London.

McColly, M. & Weier, D. (1983). Literary Attribution and Likelihood-Ratio Tests: The Case of the Middle English 'Pearl' Poems. *Computers & the Humanities*, 17, 65-75.

McKenzie, D.P. & Forsyth, R.S. (1995). Classification by Similarity: An Overview of Statistical Methods of Case-Based Reasoning. *Computers in Human Behavior*, 11(2), 273-288.

McKenzie, D.P., McGory, P.D., Wallace, C.S., Low, L.H., Copolov, D.L. & Singh, B.S. (1993). Constructing a Minimal Diagnostic Decision Tree. *Methods of Information in Medicine*, X, 1297-1302.

McLachlan, G. (1992). *Discriminant Analysis & Statistical Pattern Recognition*. Wiley, New York.

McMahon, L.E., Cherry, L.L. & Morris, R. (1978). Statistical Text Processing. *Bell System Technical Journal*, 57(6), 2137-2154.

Mendenhall, T.C. (1887). The Characteristic Curves of Composition. *Science*, 11, 237-249. [March supplement.]

Merriam, T.V.N. (1989). An Experiment with the Federalist Papers. *Computers & the Humanities*, 23, 251-254.

Merriam, T.V.N. (1992). Doctoral Thesis, Kings College, University of London.

Merriam, T.V.N. & Matthews, R.A.J. (1994). Neural Computation in Stylometry II: an Application to the Works of Shakespeare and Marlowe. *Literary & Linguistic Computing*, 9(1), 1-6.

Michie, D., Spiegelhalter, D.J. & Taylor, C.C. (1994) eds. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, Chichester.

Milic, L.T. (1967). *A Quantitative Approach to the Style of Jonathan Swift*. Mouton & Co., The Hague.

Milic, L.T. (1990). The Century of Prose Corpus. *Literary & Linguistic Computing*, 5(3), 203-208.

Mingers, J. (1987). Expert Systems -- Rule Induction with Statistical Data. *J. Operational Res. Soc.*, 38(1), 39-47.

Mingers, J. (1989a). An Empirical Comparison of Selection Measures for Decision-Tree Induction. *Machine Learning*, 3, 319-342.

Mingers, J. (1989b). An Empirical Comparison of Pruning Methods for Decision-Tree Induction. *Machine Learning*, 4, 227-243.

Morton, A.Q. (1965). The Authorship of Greek Prose. *J. Royal Statistical Society (A)*, 128(2), 169-233.

Morton, A.Q. (1978). *Literary Detection: How to Prove Authorship and Fraud in Literature and Documents*. Bowker Publishing Co.

Mosteller, F. & Tukey, J.W. (1977). *Data Analysis and Regression*. Addison-Wesley, Reading, Mass.

Mosteller, F. & Wallace, D.L. (1984). *Applied Bayesian and Classical Inference: the Case of the Federalist Papers*. Springer-Verlag, New York. [extended edition of: Mosteller & Wallace (1964). *Inference and Disputed Authorship: the Federalist*. Addison-Wesley, Reading, Massachusetts.]

Mosteller, F. & Wallace, D.L. (1972). Deciding Authorship. In: J.L. Tanur, ed. *Statistics: a Guide to the Unknown*. Holden-Day Inc., San Francisco.

Murphy, P.M. & Aha, D.W. (1991). *UCI Repository of Machine Learning Databases*. Dept. Information & Computer Sceince, University of California at Irvine, CA. [Machine-readable depository: http://www.ics.uci.edu/~mlearn/MLRepository/html.]

Murry, J. M. (1960). *The problem of Style*. Oxford Paperbacks, London. [first published 1922.]

Murthy, S., Kasif, S. & Salzberg, S. (1995). A System for Induction of Oblique Decision Trees. *Journal of Artificial Intelligence Research*, 2, 1-32.

Naylor, C. (1989). The Acquisition of Natural Language by Machine. In: R.S. Forsyth, ed., *Machine Learning: Principles & Techniques*. Chapman & Hall, London.

Oakes, M.W. (1982). *Statistical Inference: a Commentary for the Social and Behavioural Sciences*. John Wiley & Sons, Chichester.

Oakman, R.L. (1980). *Computers in Literary Research*. University of South Carolina Press, Columbia, SC.

Oliver, I. (1993). *Programming Classics: Implementing the World's Best Algorithms*. Prentice-Hall, Englewood Cliffs, NJ.

Openshaw, S. (1988). Building an Automated Modeling System to Explore a Universe of Spatial Interaction Models. *Geographical Analysis*, 20(1), 31-46.

Paechter, B. (1994). Optimizing a Presentation and Assessment Timetable using Evolutionary Algorithms. In: T.C. Fogarty, ed., *Evolutionary Computing*. Springer-Verlag, Berlin.

Palmer, F.R. (1971). *Grammar*. Penguin Books, Harmondsworth, Middlesex.

Pearce, T.S. (1967). *T.S. Eliot*. Evans Brothers, London.

Penny, W.D. (1993). *The Stoarge, Training and Generalization Properties of Multi-Layer Logical Neural Networks*. Unpublished Doctoral Thesis, Dept. Electrical Engineering and Electronics, Brunel University, Middx.

Piatetsky-Shapiro, G. & Frawley, W.J. (1991) eds. *Knowledge Discovery in Databases*. MIT Press, Cambridge, Mass.

Porat, S. & Feldman, J.A. (1991). Learning Automata from Ordered Examples. *Machine Learning*, 7, 109-138.

Porkess, R. (1988). *Dictionary of Statistics*. Collins, London.

Pound, E.L. (1977). *Selected Poems*. Faber & Faber Limited, London.

Press, W.H., Flannery, B.P., Teukolsky, S.A. & Vetterling, W.T. (1986). *Numerical Recipes: the Art of Scientific Computing*. Cambridge University Press.

Pudney, J.S. (1946). *Selected Poems*. John Lane The Bodley Head Ltd., London.

Pudney, J.S. (1967). *Spill Out*. J.M. Dent & Sons Ltd., London.

Pudney, J.S. (1969). *Spandrels*. J.M. Dent & Sons Ltd., London.

Quastler, H. (1956). *Information Theory in Psychology*. Free Press of Glencoe, New York.

Quinlan, J.R. (1982). Semi-Autonomous Acquisition of Pattern-Based Knowledge. In: D. Michie, ed., *Introductory Readings in Expert Systems*. Gordon & Breach Science Publishers, New York.

Quinlan, J.R. (1986). Induction of Decision Trees. *Machine Learning*, 1, 81-106.

Quinlan, J.R. (1987). Simplifying Decision Trees. *Int. J. Man-Machine Studies*, 27, 221-234.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California.

Quinlan, J.R. & Rivest, R. (1989). Inferring Decision Trees Using the Minimum Description Length Principle. *Information & Computation*, 80, 227-248.

Radday, Y.T. & Shore, H. (1976). The Definite Article: a Type- and/or Author-Specifying Discriminant in the Hebrew Bible. *ALLC Bulletin*, 4(1), 23-31.

Reaven, G.M. & Miller, R.G. (1979). An Attempt to Define the Nature of Chemical Diabetes using a Multidimensional Analysis. *Diabetologia*, 16, 17-24.

Rechenberg, I. (1973). *Evolutionsstrategie -- Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Halzboog, Stuttgart.

Ricks, C.B. (1984). *The Force of Poetry*. Clarendon Press, Oxford.

Ripley, B.D. (1993). Statistical Aspects of Neural Networks. In: O. Bondorff-Nielsen, D. Cox, J. Jensen & W. Kendall, eds., *Chaos & Networks: Statistical & Probabilistic Aspects*. Chapman & Hall, London.

Rissanen, J. (1987). Stochastic Complexity. *J. Royal Statistical Soc. (B)*, 49(3), 223-239.

Ritter, F.E. (1991). Towards Fair Comparisons of Connectionist Algorithms through Automatically Optimized Parameter Sets. *Proc. Annual Conf. of the Cognitive Science Soc.*, 877-881, LEA: Hillsdale, N.J.

Ritter, G.L., Woodruff, H.B., Lowry, S.R. & Isenhour, T.L. (1974). An Algorithm for a Selective Nearest Neighbour Decision Rule. *IEEE Trans. on Info. Theory*, IT-21(6), 665-669.

Rivest, R. (1987). Learning Decision Lists. *Machine Learning*, 2, 229-246.

Roberts, P.D. (1986). *How Poetry Works*. Pelican Books, Middx.

Ross, D. & Hunter, D. (1994). Micro-EYEBALL: An Interactive System for Producing Stylistic Descriptions and Comparisons. *Computers & the Humanities*, 28, 1-11.

Ryan, B.F., Joiner, B.L. & Ryan, T.A. (1985). *Minitab Handbook*, 2nd edition. Duxbury Press, Boston, Mass.

Safavian, S.R. & Landgrebe, P. (1991). A Survey of Decision Tree Classifier Methodology. *IEEE Trans. on Systems, Man & Cybernetics*, SMC-21(3), 660-674.

Sagan, C. (1981). *Cosmos*. Macdonald Futura, London.

Sahin, K. & Sawyer, K. (1989). The Intelligent Banking System: Natural Language Processing for Financial Communications. In: H. Schorr & A. Rappaport, eds. *Innovative Applications of Artificial Intelligence*, vol. 1. AAAI Press, Menlo Park, CA.

Salton, G. & McGill, M. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, N.Y.

Samuel, A.L. (1967). Some Studies in Machine Learning using the Game of Checkers, Part II. *IBM J. of Research & Development*, 11.

Schmitt, S.A. (1969). *Measuring Uncertainty: an Elementary Introduction to Bayesian Statistics*. Addison-Wesley Publishing, Reading, Mass.

Schoeneburg, E. (1994). The Genetic Formula Engine: a Unified Approach for Neural Network Optimization & Multivariate Data Analysis Based on Genetic Algorithms. In: *Adaptive Computing & Information Processing*, Unicom Seminars Ltd., Uxbridge, Middx., 413-426.

Schum, D.A. (1994). *The Evidential Foundations of Probabilistic Reasoning*. John Wiley & Sons, N.Y.

Searle, J. (1982). The Philosophy of Language. In: B. Magee, ed., *Men of Ideas*. Oxford University Press, Oxford.

Sedgewick, R. (1988). *Algorithms*, second edition. Addison-Wesley, Reading, Mass.

Sejnowski, T.J. & Rosenberg, C.R. (1987). Parallel Networks that Learn to Pronounce English Text. *Complex Systems*, 1, 145-168.

Selfridge, O.G. (1959). Pandemonium: a Paradigm for Learning. In: *Mechanization of Thought Processes*, 513-526: Proc. Symposium held at the National Physical Laboratory. HMSO, London.

Shannon, C.E. (1951). Prediction and Entropy of Printed English. *Bell System Tech. J.*, 30, 50-57.

Shannon, C.E. & Weaver, W. (1949). *The Mathematical Theory of Communication*. University of Illinois Press, Urbana.

Sichel, H.S. (1986). Word Frequency Distributions and Type-Token Characteristics. *Math. Scientist*, 11, 45-72.

Siegel, S. & Castellan, N.J. (1988). *Nonparametric Statistics for the Behavioural Sciences*, 2nd edition. McGraw-Hill, New York.

Smith, J.E., Fogarty, T.C. & Johnson, I.R. (1994). Genetic Selection of Features for Clustering and Classification. *IEE Colloquium on Genetic Algorithms in Image Processing & Vision*. London.

Smith, M.W.A. (1983). Recent Experience and New Developments of Methods for the Determination of Authorship. *ALLC Bulletin*, 11, 73-82.

Smith, M.W.A. (1985). An Investigation of Morton's Method to Distinguish Elizabethan Playwrights. *Computers & the Humanities*, 19, 3-21.

Smith, P.L. (1982). Measures of Variance Accounted for: Theory and Practice. In: G. Keren, ed., *Statistical & Methodological Issues in Psychology & Social Sciences Research*. Lawrence Erlbaum Associates, Hillsdale, N.J.

Specht, D.F. & Shapiro, P.D. (1991). Generalized Accuracy of Probabilistic Neural Networks Compared with Back-Propagation Networks. *Proc. Internat. Conf. on Neural Networks*, Seattle, WA, 887-892.

Spender, S. & Hall, D. (1970), eds. *The Concise Encyclopaedia of English and American Poets and Poetry*, 2nd edition. Hutchinson, London.

Steck, G.P. (1962). Stochastic Model for the Browning-Bledsoe Pattern Regognition Scheme. *IRE Trans. on Electronic Computers*, April, 274-282.

Stone, P.J., Dunphy, D.C., Smith, M.S. & Ogilvie, D.M. (1966). *The General Inquirer: A Computer Approach to Content Analysis in the Behavioral Sciences*. MIT Press, Cambridge, Mass.

Storer, J.A. (1988). *Data Compression: Methods & Theory*. Computer Science Press, Rockville, Maryland.

Swonger, C.W. (1972). Sample Set Condensation for a Condensed Nearest Neighbour Decision Rule for Pattern Recognition. In: S. Watanabe, ed., *Frontiers of Pattern Recognition*. Academic Press, Orlando, Florida.

Taylor, G. (1985). Shakespeare's New Poem: a Scholar's Clues and Conclusions. *New York Times Book Review*, 15th December, 11-14.

Taylor, G. (1987). The Canon and Chronology of Shakespeare's Plays. In: S. Wells & G. Taylor, eds., *William Shakespeare: a Textual Companion*. Clarendon Press, Oxford.

Thirkell, L.A. (1992). A Connectionist Approach to Authorship Determination. In: *Proc. 19th Internat. Conf. of ALLC*, at Christ Church College, Oxford University.

Thisted, R. & Efron, B. (1987). Did Shakespeare Write a Newly-Discovered Poem? *Biometrika*, 74(3), 445-455.

Thomas, D.M. (1952). *Collected Poems 1934-1952*. J.M. Dent & Sons Ltd., London.

Tomek, I. (1976). An Experiment with the Edited Nearest-Neighbour Rule. *IEEE Trans. on Systems, Man & Cybernetics*, SMC-6(6), 448-452.

Tukey, J.W. (1977). *Exploratory Data Analysis*. Addison-Wesley, Reading, Mass.

Tweedie, F., Singh, S. & Holmes, D.I. (1994). Neural Network Applications In Stylometry: the Federalist Papers. In: A.I.C. Monaghan, ed., *CSNLP*, Natural Language Group, Dublin City University.

Uhr, L. & Vossler, C. (1966). A Pattern-Recognition Program that Generates, Evaluates, and Adjusts its own Operators. In: L. Uhr, ed. *Pattern Recognition*. John Wiley & Sons, New York.

Ule, L. (1982). Recent Progress in Computer Methods of Authorship Determination. *ALLC Bulletin*, 10(3), 73-89.

Ullman, J.R. (1974). Automatic Selection of Reference Data for Use in a Nearest Neighbour Method of Pattern Classification. *IEEE Trans. on Info. Theory*, IT-20(4), 541-543.

Vadera, S. & Nechab, S. (1994). ID3, its Children, and their Safety. *British Computer Society's SGES Newsletter*, 31, 11-21.

Venkman, P.E. & Stantz, R.A. (1989). Referential Anomalies as Authorial Authenticators. *J. Applied Metalinguistics*, 1, 7-13.

Voutilainen, A., Heikkila, J. & Anttila, A. (1992). *Constraint Grammar of English: A Performance-Oriented Introduction*. Publication No. 21, Dept. General Linguistics, University of Helsinki, Finland.

Wallace, C.S. & Boulton, D.M. (1968). An Information Measure for Classification. *Computer Journal*, 11, 185-195.

Wallace, C.S. & Freeman, P.R. (1987). Estimation and Inference by Compact Coding. *J. Royal Statistical Soc. (B)*, 49, 240-265.

Wasserman, P.D. (1993). *Advanced Methods in Neural Computing*. Van Nostrand Reinhold, New York.

Webb, C. (1994). Interviewed in: *The Independent Magazine*, 5 February, page 35.

Weiss, S.M. & Kulikowski, C.A. (1991). *Computer Systems that Learn*. Morgan Kaufmann, San Mateo, CA.

Wells, S. & Taylor, G. (1987). *William Shakespeare: a Textual Companion*. Clarendon Pres, Oxford.

Wickmann, D. (1976). On Disputed Authorship, Statistically. *ALLC Bulletin*, 4(1), 32-41.

Williams, C.B. (1970). *Style and Vocabulary: Numerical Studies*. Charles Griffin & Co., London.

Wills, G. (1982) ed. *The Federalist Papers of Alexander Hamilton, James Madison & John Jay*. Bantam Books, Toronto.

Wilson, I. (1993). *Shakespeare: The Evidence*. Headline Book Publishing, London.

Wilson, R.A. (1982). *Schroedinger's Cat Trilogy*. Pocket Books, New York.

Witten, I.H., Neal, R.M. & Cleary, J.G. (1987). Arithmetic Coding for Data Compression. *Communications of the ACM*, 30(6), 520-540.

Wolff, J.G. (1975). An Algorithm for the Segmentation of an Artificial Language Analogue. *Brit. J. Psychology*, 66(1), 79-90.

Wolff, J.G. (1976). Frequency, Conceptual Structure and Pattern Recognition. *Brit. J. Psychology*, 67(3), 377-390.

Wolff, J.G. (1988). Learning Syntax & Meanings through Optimization and Distributional Analysis. In: Y. Levy, I.M. Schlessinger & M.D.S. Braine, eds., *Categories & Processes in Language Acquisition*. Lawrence Erlbaum, New York, 179-215.

Wolff, J.G. (1991). *Towards a Theory of Cognition and Computing*. Ellis Horwood, Chichester.

Wolff, J.G. (1993). Computing, Cognition & Information Compression. *AI Communications*, 6(2), 107-127.

Woolf, A.V. (1992). *The Waves*. Vintage, London. [Original edition: Hogarth Press, 1931.]

Xu, L. & Yuille, A. (1992). Robust PCA Learning Rules based on a Statistical Physics Approach. In: *Proc. Internat. Joint Conf. on Neural Networks*, Baltimore, book 1, 812-817. IEEE, New York.

Yardi, M.R. (1946). A Statistical Approach to the Problem of the Chronology of Shakespeare's Plays. *Sankhya: Indian J. Statistics*, 7(3), 263-268.

Yeats, W.B. (1961). *The Collected Poems of W.B. Yeats*. Macmillan & Co. Limited., London.

Yule, G.U. (1944). *The Statistical Study of Literary Vocabulary*. Cambridge University Press, Cambridge.

Zhu, G. & Shadbolt, N.R. (1994). Mining Knowledge: the Partial Parsing of Texts. *Proc. Third Internat. Conf. on Practical Applications of Prolog*, London.

Zimmerman, R.A. [See Dylan, B.]