FEATURE-FINDING FOR TEXT CLASSIFICATION

Richard S. Forsyth & David I. Holmes,

Bristol Stylometry Research Unit,
Department of Mathematical Sciences,
University of the West of England,
Bristol BS16 1QY, UK.

Corresponding author: Richard Forsyth.
email: forsyth_rich@yahoo.co.uk

[Cite as:
Forsyth, R.S. & Holmes, D.I. (1996). Feature-finding for text classification. *Literary & Linguistic Computing*, 11(4), 163-174.
]

"Every man's language has, first, its individualities; secondly, the common properties of the class to which he belongs; and thirdly, words and phrases of universal use." -- Samuel Taylor Coleridge (1906 [1817]).

Abstract

Stylometrists have proposed and used a wide variety of textual features or markers, but until recently very little attention has been focused on the question: where do textual features come from? In many text-categorization tasks the choice of textual features is a crucial determinant of success, yet is typically left to the intuition of the analyst. We argue that it would be desirable, at least in some cases, if this part of the process were less dependent on subjective judgement. Accordingly, this paper compares five different methods of textual feature finding that do not need background knowledge external to the texts being analyzed (three proposed by previous stylometers, two devised for this study). As these methods do not rely on parsing or semantic analysis, they are not tied to the English language only. Results of a benchmark test on 10 representative text-classification problems suggest that the technique here designated *Monte-Carlo Feature-Finding* has certain advantages that deserve consideration by future workers in this area.

Keywords: Linguistic Variables, Minimum-Deviance Classification, Monte-Carlo Methods, Pattern Recognition, Stylometry, Text Categorization.

1. Introduction

In their attempts to capture consistent and distinctive features of linguistic style, stylometrists have used a bewildering variety of textual indicators (see: Holmes, 1994). In the majority of stylometric studies, however, the choice of which indicators (or `markers') to use in a given problem is left to the discretion of the investigator (e.g. Dixon & Mannion, 1993; Matthews & Merriam, 1993; Merriam & Matthews, 1994; Holmes & Forsyth, 1995). An advantage of this practice is that it allows the exercise of human judgement, and thus can sometimes save a time-consuming search for suitable descriptors. On the other hand, it also inevitably involves subjectivity. Very often the choice of suitable linguistic markers is crucial to the development of an effective discriminant rule; but, being subjective, it may not be replicable on another problem. A further disadvantage is that each stylometrist typically has a `tool-kit' of favourite marker types which encompasses only a small fraction of those that might be used.

The situation is similar in the related fields of multivariate pattern recognition and machine learning (Everitt & Dunn, 1991; Quinlan, 1993): most studies begin by presuming that a suitable set of attributes or features has already been found. In text analysis this presumption is more than usually questionable. It is arguable, for instance, that Mosteller and Wallace (1984 [1964]), in their classic study of *The Federalist Papers*, brought a good deal of background knowledge to the task of finding features 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.

For these and other reasons, a number of studies have appeared recently (e.g. Burrows, 1992; Binongo, 1994; Burrows & Craig, 1994; Kjell, 1994; Ledger & Merriam, 1994) in which the features used as indicators are not imposed by the prior judgement of the analyst but are -- at least to a large extent -- dictated by the texts being analyzed. The main aim of the present paper is to advance this trend, by conducting a test of five different methods of textual feature-finding (three proposed by previous researchers and two newly devised) on a mixed set of text-classification problems. Although no set of textual markers can be entirely free of preconceptions, the five methods of feature-finding tested here depend only minimally on human judgement. In addition, none of them presupposes that the text being analyzed is in English.

A secondary aim is to show that categorization of quite short segments of text (shorter than most previous stylometrists have tried to classify) is feasible using a relatively simple algorithm -- provided that suitable features have been found.

Thus this paper describes an experiment with a straightforward plan: a single classification technique is applied to 10 test problems using five different types of textual marker. The main response variable is the proportion of correct classifications achieved on unseen test samples (each relatively short); the main factor of interest is the source of markers (with five levels).

Before describing these different marker sources, however, it is necessary to give a brief outline of (1) the benchmark problems, and (2) the classification algorithm used -- as both of these have novel aspects that will be unfamiliar to many readers.

2. The Bristol Benchmark Suite

In many areas of computing, benchmarking is a routine practice. For instance, when compilers are tested for compliance to a programming-language specification, it is normal to apply them to a suite of benchmark cases and record any divergence from expected behaviour.

There is not room here to go into the pros and cons of benchmarking in any depth, except to acknowledge that sets of benchmarks do have drawbacks as well as advantages -- one disadvantage being that once a a benchmark suite is widely accepted as standard, an incentive exists to optimize performance on that suite, possibly at the expense of other problem types. Nevertheless benchmarking does have a role to play in setting agreed and objective standards. For example, it is arguable that in the field of forecasting, the work of Makridakis and colleagues (e.g. Makridakis & Wheelwright, 1989), who tested a number of forecasting methods on a wide range of (mostly economic) time series, transformed the field -- leading to both methodological and practical advances.

Likewise, in machine learning, the general acceptance of the Machine-Learning Database Repository (Murphy & Aha, 1991) as a *de facto* standard, and its employment as the basis for extensive comparative tests (e.g. Michie et al., 1994) has thrown new light on the merits and demerits of various competing algorithms.

Although billion-byte public-domain archives of text exist, e.g. Project Gutenberg and the Oxford Text Archive, stylometry currently lacks an equivalent set of accepted test problems. Thus we have been forced to compile our own. For any deficiences in it, we apologize; but it is hoped that the test problems described below will evolve in due course into something of value to the field as a whole. (It is fair to state that even in its present admittedly underdeveloped form the suite of problems described below already provides a more varied and exhaustive range of tests than any previously reported in the stylometric literature.)

The text-categorization problems in this suite were selected to fulfil a number of requirements.

1.Provenance: the true category 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 applications.

The last point may need amplification. Although some huge text samples are available, most text-classification tasks in real life require decisions to be made on the basis of samples in the order of thousands or tens of thousands, rather than hundreds of thousands or millions, of words. An enormous training sample of undisputed text is, therefore, something of a luxury. It was felt important that the method used here should be able to perform reasonably well without reliance on this luxury.

Subject to these constraints, ten test problems were chosen. This suite (called here Tbench95) contains three authorship problems, three chronology problems, two content-based problems, and two synthetic quasi-random problems. Fuller information is given in Appendix A.

3. Pre-Processing & Other Preliminaries

In order to impose 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 text samples were passed through a program called PRETEXT before being analyzed. This program makes some minor formatting changes: tabs and other white-space characters are converted into blanks; runs of multiple blanks are converted into single blanks; and upper-case letters are converted into lower case. By far the most important change made by PRETEXT, however, is to break running text into segments that are then treated as units or cases to be classified.

Just what consititutes a natural unit of text is by no means obvious. Different researchers have made different decisions about the best way of segmenting long texts and thus turning a single sequence of characters into a number of cases or observations. Some have used fixed-length blocks (e.g. Elliott & Valenza, 1991); others have respected natural subdivisions in the text (e.g. Ule, 1982). Both approaches have merits and drawbacks.

Because linguistic materials have a hierarchical structure there is no universally correct segmentation scheme. Textual units could range from single lines or sentences at one end of the scale to chapters or even whole books at the other. Generally speaking, smaller text units are too short to provide opportunities for stylistic habits to operate on the arrangement of internal constituents, while larger units are insufficiently frequent to provide enough examples for reliable statistical inference.

The compromise adopted for the present study was to break all texts into blocks of roughly the same length. In fact, each block boundary was taken as the first new-line in the original text on or after the 999th byte in the block being formed. As a result, mean block size is always between 1010 and 1030 characters. Such units will be referred to as kilobyte lines.

The number of words per kilobyte line varies according to the type of writing. A representative figure for Tbench95 as a whole is 187 words per line. This is, in fact, the median word length per line of the five files nearest to the median overall line length (of 1018 bytes). The exact figure is unimportant, though it should be noted that each kilobyte line (almost invariably less than 200 words) is very short in comparison with the size of text units that previous stylometrists have felt worth analyzing. In other words, this is an attempt to work with text units near the lower limit of what has thus far been considered feasible. Evidence of this is provided by the two quotations below, made 20 years apart.

"It is clear in the present study that there is considerable loss in discriminatory power when samples fall below 500 words". (Baillie, 1974)

«Feature-Finding for Text Classification»

"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)

Further confirmation is provided in Table 1, which collates information from a selection of stylometric studies, showing the length, in words, of text blocks that various researchers have tried 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. Rows have been arranged in descending order of this norm. The Range column gives the sizes, again in words, of the smallest and largest text block 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 segment ever given an attribution (so far as we know) is 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.

Table 1 -- Size of Text Blocks Analyzed by Various Stylometrists.

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)	Bronte sisters	4000	500-8000
Milic (1967)	Jonathan Swift	3500	3324-3777
Mosteller & Wallace (1984)	Federalist papers	2100	906-3506
Ledger (1989)	Platonic dialogues	1000	1000-1000
Matthews & Merriam (1993)	Elizabethan drama	1000	[unknown]
Binongo (1994)	Nick Joaquim's short stories	1000	[unknown]
Elliott & Valenza (1991)	Elizabethan poetry	500	425-500
Thisted & Efron (1987)	Shakespeare & contemporaries	400	234-495

It is very rare for anyone to attempt to classify a text segment of less than 250 words; so the problem of classifying kilobyte-sized chunks, averaging around 187 words and almost always less

than 200 words, must be regarded as a relatively stringent stylometric test. A method that is successful under these conditions thus deserves serious consideration.

4. Classification by Minimum Deviance

The particular classification algorithm used is not the prime concern of this paper. For that reason it was decided to employ a relatively simple method for the trials reported here. We term this the `method of minimum deviance'. It is a variant of the nearest-centroid classifier, which is, in turn, related to the well-known and popular nearest-neighbour classifier (1-NNC): see, for example, Dasarathy (1991). Similarity-based methods of this general type have been shown in several empirical trials to be surprisingly robust (Forsyth, 1990; Aha et al., 1991; Michie et al., 1994; McKenzie & Forsyth, 1995). Moreover, such methods are easy to understand and to implement.

Basically, the minimum-deviance classifier, as implemented here, uses a training set of examples with known class membership to compute a centroid (multi-dimensional average) for each category. Then on a fresh or unseen case a measure of distance from (or equivalently, proximity to) each class centroid is computed and the current case is assigned to the category of the centroid which it most resembles. Most such algorithms use a Euclidean or City-block distance metric, but in the present case the `distance' measure used is termed deviance. It is computed as follows

$$Deviance(i,c) = \sum_j \frac{(x_{ij} - m_{cj})^2}{(m_{cj} + 1.0)}$$

where i is the current case, j is a feature index, c is a class code, and m_{cj} is the mean value of class c on feature j in the training set. If the features are, for example, word frequencies then x_{ij} is simply the number of times that word j occurs in line i. (This relies on using lines, or blocks, of approximately equal length, as is done here.)

This measure is asymptotically related to the Chi-squared statistic

$$\chi^2 = \sum \frac{(o - e)^2}{e},$$

with m_{cj} being the expected value under the hypothesis that the instance belongs to category c. The +1.0 in the denominator can be seen as a slight bias, downgrading the effect of infrequently occurring features, as well as avoiding division by zero.

Overall, minimum-deviance classification is a simple, fast and intuitively appealing technique which appears to give good results.

5. Data-Driven Feature-Finding

In this section we return to the main focus of the investigation by outlining three of the methods of data-driven feature-finding tested on the benchmark suite, namely those proposed by previous

researchers. (The two novel methods need somewhat fuller discussion, and are covered in the next three sections.)

1.Letters (Ledger & Merriam, 1994);
2.Most Common Words (Burrows, 1992);
3.Digrams (Kjell, 1994).

The first and simplest method is simply to treat each letter of the alphabet as a feature, i.e. to count the frequency of each of the 26 letters in each kilobyte line. At first glance this would seem not just simple, but simplistic. However, several previous studies -- most notably Ledger & Merriam (1994), but also Ule (1982) and Ledger (1989) -- have reported surprisingly good results when using letter-counts as stylistic indicators. At the very least, it allows us to establish a baseline level of performance: more sophisticated features sets need to outperform letter counting to justify their added complexity.

The second type of textual feature has been used by Burrows (1992) as well as Binongo (1994), among others, not only in authorship attribution but also to distinguish among genres. Essentially it involves finding the most frequently used words and treating the rate of usage of each such word in a given text as a feature. The exact number of common words used varies by author and application. Burrows and colleagues (Burrows, 1992; Burrows & Craig, 1994) discuss examples using anywhere from the 50 most common to the 100 most common words. Binongo (1994) uses the commonest 36 words (after excluding pronouns). Greenwood (1995) uses the commonest 32 (in New Testament Greek). In the present study, the most frequent 96 words in the combined training samples were used -- without exclusions. Most such words are function words, and thus this approach can be said to continue the tradition, pioneered by Mosteller & Wallace (1984 [1964]), of using frequent function words as markers.

The third method tested here uses digram counts as features. Kjell (1994) reported good results in assigning *Federalist* essays written either by Hamilton or Madison to the correct authors using a neural-network classifier to which letter-pair frequencies were given as input features. The present method is a slight generalization of Kjell's in that it uses character pairs rather than just letter pairs; so, for instance, digrams involving blanks or other punctuation marks may be used. Another difference from Kjell's work is that, instead of selecting letter pairs for their discriminatory ability, the 96 commonest digrams in the combined training sets of the problem concerned were used, thus rendering this approach more directly comparable with that of Burrows.

Note that all three methods of feature-finding outlined above share four desirable properties: (1) they are easy to compute; (2) they are easy to explain; (3) they are interlingual, i.e. they are not limited to English; (4) they require no exercise of skill by a user but can be found quite automatically. These four properties also apply to the next two methods.

6. Progressive Pairwise Chunking

To broaden the scope of this comparison somewhat, two novel feature-finding techniques were also tested.

The first of these is here called progressive pairwise chunking. It attempts to avoid the artificiality of always using fixed-length markers (such as digrams or trigrams) while also allowing marker substrings that are shorter than words (e.g. an affix such as `ed ') or which cross word boundaries (e.g. a collocation such as `in the'). This is done by adapting a method first described (in different contexts) by Wolff (1975) and Dawkins (1976).

Essentially the algorithm scans a byte-encoded text sequence, looking for the most common pair of symbols. At the end of each scan it replaces all occurrences of that pair by a newly allocated digram code. This process is repeated for the next most common pair and so on, till the requested number of pairings have been made. The program used here assumes that character codes from ASCII 128 upwards are free for reassignment (as is the case with Tbench95), so byte codes from 128 onwards are allocated sequentially. Its output is a list of doublets. These are not always digrams, since previously concatenated doublets can be linked in later passes. Thus the program can build up quite long chains, if they occur in the data -- identifying sequential dependencies of quite a 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 transition probabilities spanning more than a few items.

An extract from this program's output when applied to the NAMESAKE training data (containing poems by Bob Dylan and Dylan Thomas) is shown as Table 2. For the sake of brevity, only the most common 16 substrings are shown, plus a selection of less common strings that illustrate the potential of this method.

Table 2 -- Example of Substrings Formed by Progressive Pairwise Chunking.

```
FREQLIST output;  date: 01/04/96 12:52:30
1  C:\BM95\BD.TRN
68016 bytes.
2  C:\BM95\DT.TRN
45952 bytes.
113968 bytes.
2 input files read.

Most frequent markers :
   1 `e `                    4030
   2 ` t`                    3501
   3 `th`                    2994
   4 ` th`                   2482
   5 `s `                    2122
   6 ` a`                    2075
   7 ` the`                  2042
   8 `d `                    1963
   9 `in`                    1814
  10 ` s`                    1772
  11 `t `                    1767
  12 ` the `                 1668
  13 `, `                    1637
```

```
 14 ` i`                    1500
 15 ` w`                    1486
 16 `an`                    1412

 29 `and `                   742

 32 ` and `                  700

 41 `ing`                    637
 42 `you`                    623
 43 ` you`                   618

 46 `ed `                    590

 60 ` to `                   417

 62 `wh`                     389

 76 `'s `                    293
 77 `e, `                    293

 80 `s, `                    273
 81 ` the s`                 269

 90 `in the `                210

 96 `ver`                    159
```

It will be seen that, as well as pure digrams, this method tends to find common trigrams (e.g. `ver'), words (e.g. ` the '), morphemes (e.g. `ing', `'s ') and collocations (e.g. `in the'). Some of the other substrings, such as `the s', do not fall naturally into any pre-existing linguistic grouping. As it turns out, Dylan Thomas is rather fond of following the definite article with the letter `s' -- a fact that more conventional methods of feature finding would not be able to exploit.

7. Monte-Carlo Feature-Finding

The fifth and last method of feature-finding tested in this study takes the idea implicit in the progressive chunking method (that it is desirable to employ a variety of marker substrings, both longer and shorter than words) to what may be thought its logical conclusion. Monte-Carlo Feature-Finding is simply a random search for substrings that exist in the training data. Here this process is implemented by a program called CHISUBS. This finds textual markers (short substrings) without any guidance from the user, merely by searching through a given set of training texts.

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. Chi-

squared is used as an index of distinctiveness here because McMahon et al. (1978) used it succesfully for a similar purpose -- expected values being calculated on the basis of equal rates of usage.

In the experiments reported here, the values for the parameters mentioned above were as follows: L=7, N=3600, C=96. Thus the program sought 3600 different substrings, of from 1 to 7 characters long, and then retained only the most discriminating 96 of them[1].

Table 3 shows the result of running CHISUBS on training data from the *Federalist Papers* (see Appendix A for details). It illustrates the sort of markers found by this process. Only the best 26 markers have been listed, to save space.

Table 3 -- Marker Substrings Derived from FEDS Data.

```
CHISUBS output;  date: 01/04/96 13:35:23
gramsize = 7
1  C:\BM95\HAMILTON.TRN
224555 bytes.
2  C:\BM95\MADISON.TRN
232931 bytes.
proportion in class 1 = 0.490845085
proportion in class 2 = 0.509153822
Grams kept = 96

Rank         Substring  Chi-score    Frequencies
1                `pon`  90.7905528    141.  22.
2             ` would`  69.4649456    334.  158.
3             `there `  68.7778962    137.  32.
4               ` wou`  67.7237888    334.  160.
5               ` on `  67.0970655    119.  293.
6            ` would `  64.6422832    322.  155.
7                `up`  61.7127024    292.  137.
8                `na`  58.418292    693.  455.
9              `owers`  56.8864289    30.  128.
10           `partmen`  56.2652439    12.  90.
11              `wers`  51.6241599    34.  129.
12               `epa`  51.3068526    31.  123.
13              `ould`  49.8865689    461.  282.
14               `oul`  49.3130926    461.  283.
15           ` on the`  47.318841    52.  155.
16             `ould `  46.4552952    446.  276.
17               ` on`  44.845061    288.  489.
18             ` form`  44.3970607    45.  139.
```

[1] Strictly speaking the program only dumps **up to** C substrings, since those with a Chi-squared score less than K, the number of categories, are not kept; furthermore, substrings occurring less than 10 times in the combined training text are also dropped. However, none of the 10 benchmark sets used here actually gave rise to fewer than 96 markers.

```
19              `court`  42.6803592    72.  13.
20                 `wo`  42.6386397    399.  245.
21            `powers `  40.9096955    22.  93.
22            `governm`  40.6851794    149.  291.
23                 `ou`  37.9349158    1285.  1031.
24            `ernment`  37.2283842    154.  291.
25             ` there`  36.5487883    160.  72.
26              `presi`  36.2364995    67.  14.


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

To interpret this listing, it should be noted that, in the last two columns, the frequency of usage in Hamilton's training sample comes before the frequency in Madison's. As both authors are represented by almost the same amount of text, this can be read as saying that, for example, ` would' is a Hamilton marker (334 : 158) while ` on the ' is a Madison marker (52 : 155). Thus even this simple printout provides some interesting information -- although many of these items appear to be what Mosteller & Wallace (1984) would call "dangerously contextual".

Whether such reliance on contextual or content-bearing linguistic items is a weakness can be answered by empirical testing; but CHISUBS also suffers from two structural faults which, unless corrected, would detract from its appeal even if such markers prove effective in practice. Firstly, as can easily be seen above, there is plenty of redundancy (e.g. ` would' as well as ` would '). Secondly, these text fragments are often segmented at what seem to be inappropriate boundary points (e.g. `governm', which surely ought to be ` government'). The most notable example of improper fragmentation in Table 3 is `pon', which anyone who has studied the *Federalist* problem will immediately realize is an imperfect surrogate for ` upon'.

8. How Long is a Piece of Substring?

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 it often produces substring markers which, to a user, appear to be truncated at inappropriate places. Examples of this problem are `rpus' instead of `corpus' (from the MAGS data) and, most irritatingly, `pon' instead of ` upon' from the *Federalist* samples.

So the program described here was modified to alleviate 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 revised program, TEFF (Text-Extending Feature-Finder), picks short substrings at random by the same method as described, but each substring is `stretched' as much as compatible with the data as soon as it is generated and before being saved for evaluation.

The idea 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 TEFF employs 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[2] 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' were found in the Hamilton sample, as it most probably would be, then a common predecessor/successor would only be sought within that sample.

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 specimen of the results of applying this procedure to a list of substrings produced by CHISUBS from the *Federalist* data is given below as Table 4. 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,

```
27  `deraci` : ` confederacies`
```

[2] Originally `invariably preceded' (or followed) meant exactly that, but the process was rather slow, so current versions of this procedure actually stop looking after 39 consecutive occurrences of the same predecessor or successor. This does not affect substrings that occur less than 39 times, of course; and appears to make little difference to the rest.

means that the 27th item was derived from the substring `deraci' which turned out always to be embedded within the longer string ` confederacies'. From this listing it hoped that readers will be able to appreciate how the program works and judge its effectiveness.

Table 4 -- Examples of `Stretched' Substrings from Federalist Text.

```
1  `upo` : ` upon `
2  `pon` : `pon`
3  ` would` : ` would `
4  `there ` : ` there `
5  ` on ` : ` on `
6  `up` : `up`
7  `na` : `na`
8  `owers` : `powers`
9  `partmen` : ` department`
10  `wers` : `wers`
11  `epa` : `epa`
12  `ould` : `ould `
13  ` on the` : ` on the`
14  ` on` : ` on`
15  ` form` : ` form`
16  `court` : ` court`
17  `wo` : `wo`
18  `powers ` : `powers `
19  `overnme` : ` government`
20  `ou` : `ou`
21  `ernment` : `ernment`
22  ` there` : ` there`
23  `presi` : `preside`
24  ` cour` : ` cour`
25  `nat` : `nat`
26  `nmen` : `nment`
27  `deraci` : ` confederacies`
28  `dicia` : ` judicia`
29  `he stat` : ` the stat`
30  `heir` : ` their `
31  `ed` : `ed`
32  `cour` : `cour`
33  `feder` : `federa`
34  `nst` : `nst`
35  `onsti` : ` constitu`
36  `ve` : `ve`
37  `e t` : `e t`
38  `, would` : `, would `
39  `pa` : `pa`
40  `ep` : `ep`
41  `dep` : `dep`
42  `d by` : `d by `
43  `ongres` : ` congress`
44  `e` : `e`
45  `the na` : ` the nat`
```

```
46  `stituti` : `stitution`
47  `xecutiv` : ` executive`
48  ` by ` : ` by `
49  ` govern` : ` govern`
50  `execu` : ` execut`
```

It is hoped that readers will agree that expansions such as `partmen' to ` department', `dicia' to ` judicia', `he stat' to ` the stat', `ongres' to ` congress' and `heir' to ` their ' represent gains in clarity.

TEFF, however, cannot eliminate short and apparently unsuitable substrings altogether. A case in point 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 even when this was corrected `earn' was 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 and slower than TEFF and would require a good deal of fine tuning, so it has been left as a future development. Meanwhile, the results quoted in the following section were obtained with TEFF, which does, despite its simplicity, offer an improvement in intelligibility over the basic Monte-Carlo feature-finder (CHISUBS).

9. Results

To recapitulate, textual features found by five different methods were tested on a range of text-categorization problems. These five methods will be referred to as shown in Table 5.

Note that only the last type of marker is selected according to distinctiveness: the rest are chosen solely by frequency.

Table 5 -- Types of Textual marker Tested.

Code Number	Name	Brief Description
0.	LETTERS	26 letters of the Roman alpahbet
1.	WORDS	Most frequent 96 words
2.	DIGRAMS	Most frequent 96 digrams
3.	DOUBLETS	Most frequent 96 substrings found by progressive pairwise chunking
4.	STRINGS	Most distinctive 96 substrings

		found by TEFF program

Thus this experiment has a simple 2-factorial design, with five levels on the first factor (source of text markers) and 10 levels on the second (problem number). The latter is essentially a nuisance factor: the problems do differ significantly in difficulty, but this is of no great interest. The main response variable measured is the percentage of correct classifications made **on unseen test data**. Mean values are shown in Table 6.

Table 6 -- Mean Success Rates on Test Data.

Source of Textual Marker	Mean Percentage Success Rate
LETTERS	69.03
WORDS	72.96
DIGRAMS	74.18
DOUBLETS	74.87
STRINGS	79.39

These results appear in increasing order of accuracy, averaged over the 10 test problems. It would seem that LETTERS are less effective than the middle group of WORDS, DIGRAMS and DOUBLETS, while STRINGS are more effective. To test this interpretation, a 2-way Analysis of Variance on these percentage scores was performed.

As expected, there was a very highly significant main effect of problem ($F_{9,36} = 42.13$, $p < 0.0005$). Clearly some problems are harder than others. (In fact, NAMESAKE proved the easiest, with a mean success rate overall of 93.77%; while AUGUSTAN was the most difficult, with a mean success rate of 48.58%.) More interestingly, there was also a highly significant main effect of marker type ($F_{4,36} = 5.40$, $p = 0.002$). In other words, even after allowing for differences between problems, the Null Hypothesis that all five marker types give equal success rates must be rejected. (This design doe not permit testing for an interaction effect.)

To investigate the factor of marker type further, the effect of differential problem difficulty was removed by performing a 1-way Analysis of Variance not on the raw percentage success rates but on the deviations of each score from the mean for that problem, i.e. on the residuals. Once again this revealed a highly significant effect of marker type ($F_{4,45} = 6.75$, $p < 0.0005$). In addition, Dunnett's method of multiple comparisons with a standard was performed (Minitab, 1991). For this purpose, the success rate of DIGRAMS (the marker type giving the median mean score) was taken as a norm. Using a `family error-rate' of 0.05 (i.e. with a 5% significance level overall) gave an adjusted error rate of 0.0149. At this level, scores obtained by LETTERS were significantly different from those obtained by DIGRAMS (lower), as were scores obtained using STRINGS (higher). The other two marker types (WORDS and DOUBLETS) did not differ significantly from DIGRAMS in effectiveness.

Thus the appearance of a middle group consisting of WORDS, DIGRAMS and DOUBLETS than which LETTERS give significantly worse results and STRINGS significantly better was confirmed. This is perhaps unsurprising -- at least with benefit of hindsight -- given the fact that LETTERS implies using fewer attributes than the rest (26 versus 96) and that STRINGS are preselected for distinctiveness whereas the other types of marker are selected only according to frequency. None the less, the fact that this preselection did not seem to give rise to overfitting was by no means a foregone conclusion.

In a further attempt to shed light on the effect of marker type as well problem type, indicator variables (with values 0 or 1) were created and used in a regression analysis, with percentage success rate again used as the dependent variable. Five of these binary variables indicated the presence or absence of a particular type of marker (as above); four of them indicated the type of problem (namely, Authorship, Chronology, Subject-Matter or Random Data). In addition, as the number of categories is clearly a determinant of how difficult any classification task is, the reciprocal of the number categories was also computed (1/k) and used as a predictor variable. This variable gives the proportion of correct classifications expected by chance, assuming equal prior probabilities.

These ten variables (expected chance success-rate, five indicators of marker type and four indicators of problem type) were then supplied to MINITAB's Stepwise Regression procedure as explanatory variables for predicting success rate. The procedure halted after inclusion of three variables, giving the regression equation below, with an overall R-squared of 0.7107.

Percent = 40.21 - 31.3*Rand + 83*Invcats + 6.6*Strings

This need not be taken too seriously as a predictive formula. What it does reveal, however, is that only three of these variables were worth using (between them accounting for 71.07% of the variance in classification success rate). These were, in order of inclusion:
Rand : whether the problem used random data or not;
Invcats : the reciprocal of the number of categories;
Strings : whether STRINGS were used as markers.

This can be interpreted as saying that, other factors being held constant: the random problems give success rates on average 31.3% below other problems; TEFF strings give success rates 6.6% above the expectation for other marker types; while 83/k is an estimate of the effect of varying k, the number of different categories, on percentage of correct classifications. For instance, using TEFF substrings on a non-random problem with only 2 categories the success rate predicted by this formula is 88.31%

Of course the range of problems and markers used here is restricted, so no great weight should be given to the precise values of these regression coefficients; but this analysis does suggest that while random problems are much harder than the rest, the difference in difficulty between authorship, chronology and content-based classification problems is relatively minor. It also confirms the small but significant superiority of STRINGS found by TEFF (the extended Monte-Carlo Feature-Finder) as compared with the other marker types.

10. Discussion

In this study we have attempted to examine the question of where stylometric indicators come from, a question that has, broadly speaking, only been answered implicitly by previous stylometric researchers. In doing so, we have arrived empirically at a preliminary `pecking order' among stylistic marker types based on results obtained on a benchmark suite of text classification problems. This ranking suggests that letter frequencies are less effective than other sorts of textual markers which share with letters the desirable properties of being easy to compute and applicable to a range of languages, such as common words or digrams. Thus researchers who employ letters in preference to common words or digrams in future stylometric studies may well be wasting information.

The two novel types of textual marker tested in this paper performed creditably. We propose that they both merit serious consideration by future researchers in this area.

Doublets obtained by progressive pairwise chunking would appear to be at least as informative as common words or digrams. Strings obtained by Monte-Carlo Feature-Finding gave significantly better results than the other types. While this study is limited in scope, our results do suggest that the emphasis on the **word** as the primary type of linguistic indicator may be counter-productive.

Of course this conclusion only carries force to the extent that the benchmark suite used here (Tbench95) is adequate, and it can only be regarded as a prototype. Nevertheless the very idea of using an agreed suite of benchmark problems to help provide an empirical perspective on various aspects of text classification is in itself, we believe, a contribution to stylometry; and while Tbench95 is admittedly imperfect, it provides a starting point for future developments.

We have also demonstrated here the feasibility of successfully classifying quite short text segments -- kilobyte lines, well under 200 words long on average. This we hope will encourage further work on the precise relationship between likely classification accuracy and size of text unit. To date, so it would appear from Table 1, stylometrists have `played safe' and used rather long stretches of text -- longer than strictly necessary. It would be useful to have more empirical data on this issue, to help future workers make an informed trade-off between size and accuracy. The present study at least provides a data point away from the norm.

Much else remains to be done. Two avenues which we intend to follow concern: (1) combining lexical markers such as used here with syntactic markers (as used, for instance, by Wickmann, 1976) and/or semantic markers (as used, for instance, by Martindale & McKenzie, 1995); (2) investigating interaction effects between problem type (e.g. authorship versus chronology) and marker type. A more extensive benchmark suite might allow investigation of whether certain types of marker are better with certain types of problem. For example: are common words better in authorship studies while TEFF markers work better on content-based discriminations? There was some suggestion of this in the figures obtained here, but the results are inconclusive. An extended benchmark suite would permit serious investigation of such questions.

Finally, more work is needed on feature-selection as well as feature-finding. By most standards, 96 is rather too many variables for convenience. (Even 26 variables would be considered a large feature set in some quarters.) Certainly, just showing a user a list of 96 textual features, even if they are presented in order of distinctiveness, is unlikely to promote deep insight into the nature of the differences between the text types being studied. If, however, equivalent (or better) performance could be achieved with a reduced subset of markers then the development of an accurate classifier using them would have the beneficial side-effect of promoting deeper insight into the data -- possibly an even more valuable outcome than just having a good classification rule. Initial experiments to this end, using a simple stepwise forward-selection procedure, have proved disappointing. It remains to be seen whether it is intrinsic to the nature of text that accurate classification requires the use of many indicators or whether a more efficient variable-selection algorithm, such as a genetic algorithm or simulated annealing (Siedlecki & Sklansky, 1988; Reeves, 1995), would eliminate this problem.

Goldberg (1995) has argued that textual features have some inherent characteristics -- infrequency, skewed distribution, and high variance -- which together imply that simple yet robust classification rules using just a handful of descriptors will seldom if ever be found for linguistic materials. This is an interesting conjecture, which we believe is worth attempting to falsify. A variable-selection program, which could reduce the number of textual features needed for successful text classification from around 100 to less than 20 would settle this question. It would also be a useful text-analytic tool. Moreover, a serious attempt to develop such a tool, even if it ended in failure, would have interesting implications, since it would tend to corroborate Goldberg's thesis. We hope to pursue this line of investigation in future studies.

Appendix A : Details of Benchmark Data Sets

The 10 text-classification problems that constitute TBench95 (Text Benchmark Suite, 1995 edition) form an enhanced version of the test suite used by Forsyth (1995). They also constitute a potentially valuable resource for future studies in text analysis. Collecting and editing Tbench95 has been an arduous chore, but enhancing and maintaining it could become a full-time job. Already some of the problems of corpus management (Aijmer & Altenberg, 1991) have presented themselves. It is hoped that support to overcome such problems may in due course be forthcoming, so that successors to Tbench95 may offer a genuine resource to the research community. Ideally they could be made publicly available, e.g. on the Internet, for comparative studies; but, as some of the original texts used are still under copyright, the best way of doing this will require legal advice.

Summary information about the selection of works from various authors and subdivision into training and test files is contained in section 2. Here this is amplified by giving further details concerning the sources and sizes of the texts used in the benchmark suite.

NOTE: A policy adhered to throughout was never to split a single work (article, essay, poem or song) between training and test sets.

A.1 Sources of Benchmark Data

Authorship

NAMESAKE (4 classes): Poetry by Bob Dylan and Dylan Thomas.

Songs by Bob Dylan (born Robert A. Zimmerman) were obtained from *Lyrics 1962-1985* (Dylan, 1994). In addition, two tracks from the album *Knocked Out Loaded* (Dylan, 1988) and the whole A-side of *Oh Mercy* (Dylan, 1989) were transcribed by hand and included, to give fuller coverage. An electronic version of *Lyrics 1962-1985* is apparently available from the Oxford Text Archive, but this was not known until after this selection had been compiled. Further information about the Oxford Text Archive can be obtained by sending an electronic mail message to ARCHIVE@vax.oxford.ac.uk

Poems of Dylan Thomas were obtained from *Collected Poems 1934-1952* (Thomas, 1952) with four more early works added from *Dylan Thomas: the Notebook Poems 1930-1934* (Maud, 1989). Most were typed in by hand.

EZRA (3 classes): Poems by Ezra Pound, T.S. Eliot and William B. Yeats -- three contemporaries who influenced each other's writings. For example, Pound is known to have given editorial assistance to Yeats and, famously, Eliot (Kamm, 1993).

A random selection of poems by Ezra Pound written up to 1926 was taken from *Selected Poems 1908-1969* (Pound, 1977), and entered by hand. It was supplemented by

random selection of 18 pre-1948 *Cantos*, obtained from the Oxford Text Archive. Poems by T.S. Eliot were from *Collected Poems 1909-1962* (Eliot, 1963), scanned then edited by hand. A random selection of 148 poems by W.B. Yeats was taken from the Oxford Text Archive. For checking purposes *Collected Poems* (Yeats, 1961) was used.

As is usual in machine learning the data was divided into training and testing sets. In the above cases this division was made by arranging each author's **files** in order and assigning them alternately to to test or training sets. 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 purely random allocation of individual poems to test and training sets would have been.

FEDS (2 classes): A selection of papers by two *Federalist* authors, Hamilton and Madison. This celebrated, and difficult, authorship problem -- subject of a ground-breaking stylometric analysis by Moseller & Wallace (1984 [1964]) -- is possibly the best candidate for an accepted benchmark in this field.

An electronic text of the entire Federalist papers was obtained by anonymous ftp from Project Gutenberg at
GUTNBERG@vmd.cso.uiuc.edu
For checking purposes the Dent Everyman edition was used (Hamilton et al., 1992 [1788]). Here the division into test and training sets was as follows.

Author	Training paper numbers	Test paper numbers
Hamilton	6, 7, 9, 11, 12, 17, 22, 27, 32, 36, 61, 67, 68, 69, 73, 76, 81	1, 13, 16, 21, 29, 30, 31, 34, 35, 60, 65, 75, 85
Madison	10, 14, 37-48	49-58, 62, 63

Thus, for Madison, all undisputed papers constitute the training set while the `disputed' papers constitute his test sample. This implies that we accept the view expounded by Martindale & McKenzie (1995), who state that: "Mosteller and Wallace's conclusion that Madison wrote the disputed Federalist papers is so firmly established that we may take it as given." For Hamilton, a random selection of 17 papers was chosen as a 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.

Chronology / English Poetry

ED(2 classes): Poems by Emily Dickinson, 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. The work included was all of *A Choice of Emily Dickinson's Verse*

selected by Ted Hughes (Hughes, 1993) as well as a random selection of 32 other poems from the *Complete Poems* (edited by T.H. Johnson, 1970). Data was entered by hand.

JP(3 classes): Poems by John Pudney, divided into three classes. The first category came from *Selected Poems* (Pudney, 1946) and *For Johnny: Poems of World War II* (Pudney, 1976); the second from *Spill Out* (Pudney, 1967) and the third from *Spandrels* (Pudney, 1969). All poems in these four books were used.

John Pudney (1909-1977) described his career as follows: "My poetic life has been a football match. The war poems were the first half. Then an interval of ten years. Then another go of poetry from 1967 to the present time" (Pudney, 1976). Here the task is to distinguish his war poems (published before 1948) from poems in two other volumes, published in 1967 and 1969 -- i.e. there are three categories.

WY(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. Same source as in EZRA, above.

For these problems the classification objective was to discriminate between early and late works by the same poet. The division into test and training sets for Emily Dickinson and Yeats 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 in the sequence) to two sets. The larger of the two resulting files was designated as training and the smaller as test file. With John Pudney, each book was divided into test and training sets by random allocation of individual poems.

Subject-Matter

MAGS (2 classes): This used articles from two academic journals *Literary and Linguistic Computing* and *Machine Learning*. The task was to classify texts according to which journal they came from. In fact, each `article' consisted of the Abstract and first paragraph of a single paper. These were selected by taking a haphazard subset of the volumes actually present on the shelves in UWE's Bolland Library on two separate days, then scanning in (photocopies of) the relevant portions and editing them. The results were as follows.

Literary & Linguistic Computing			Machine Learning		
Year	Articles	Words	Year	Articles	Words
			1987	1	229
			1988	1	200
			1989	1	235
1990	24	6629	1990	15	4069
1991	17	4715	1991	12	3598
1992	15	4489	1992	27	7788
1993	12	3331	1993	6	2041
1994	5	1152	1994	4	1369
1995	2	547	1995	2	530

For both journals the articles from even years were used as the training sample and those from odd years the test sample.

TROY (2 classes): Electronic versions of the complete texts of Homer's *Iliad* and *Odyssey*, both transliterated into the Roman alphabet in the same manner, were kindly supplied by Professor Colin Martindale of the University of Maine at Orono. Traditionally each book is divided into 24 sections or `books'.

For both works the training sample comprised the odd-numbered books and the test sample consisted of the even-numbered books. The task was to tell which work each kilobyte line came from.

Randomized Data

AUGUSTAN (2 classes): The Augustan Prose Sample donated by Louis T. Milic to the Oxford Text Archive. For details of the rationale behind this corpus and its later development, see Milic (1990). This data consists of extracts by many English authors during the period 1678 to 1725. It is held as a sequence of records each of which contains a single sentence. Sentence boundaries as identified by Milic were respected.

To obtain test and training sets a program was written to allocate sentences at random to four files. The larger two of these were then treated as training sets, the smaller two as test data.

RASSELAS (2 classes): The complete text of *Rasselas* by Samuel Johnson, written in 1759. This was obtained in electronic form from the Oxford Text Archive. For checking purposes, the Clarendon Press edition was used (Johnson, 1927 [1759]). This novel consists of 49

chapters. These were allocated alternately to four different files. Files 2 and 4 became the training data; files 1 and 3 were used as test sets.

The inclusion of random or quasi-random data may need a few words in justification. The chief objective of doing so here, was to provide an opportunity for what statisticians call overfitting to manifest itself. If any of the approaches tested is prone to systematic overfitting -- in the sense of exploiting random peculiarities in the training data -- then this last pair of problems will tend to reveal it. A success rate significantly **below** chance expectation on either of these data sets would be evidence of overfitting. Our view is that, as a general rule, some `null' cases should form part of any benchmark suite: as well as finding what patterns do exist, a good classifier should avoid finding patterns that don't exist.

A.2 Sizes of Benchmark Problems

Problem	Categories	Kilobytes (training, test)
DYLANS	Bob Dylan Dylan Thomas	67, 65 45, 41
EZRA	Ezra Pound T.S. Eliot W.B Yeats	80, 79 49, 40 109, 86
FEDS	Alexander Hamilton James Madison	218, 148 227, 140
ED	Emily Dickinson to 1863 after 1863	28, 23 23, 17
JP	John Pudney: War Poems Poems from Spill Out Poems from Spandrels	16, 10 17, 15 22, 20
WY	W.B. Yeats to 1914 after 1915	61, 34 52, 47
MAGS	Lit. & Ling. Computing Machine Learning	78, 54 88, 44
TROY	Iliad Odyssey	346, 308 251, 255
AUGUSTAN	Random sentences Random sentences	121, 104 108, 108
RASSELAS	Even-numbered chapters Odd-numbered chapters	60, 45 52, 48

References

Aha, D.W., Kibler, D. & Albert, M.K. (1991). Instance-Based Learning Algorithms. *Machine Learning*, 6, 37-66.

Aijmer, K. & Altenberg, B. (1991) eds. *English Corpus Linguistics*. Longman, London.

Baillie, W.M. (1974). Authorship Attribution in Jacobean Dramatic Texts. In: J.L. Mitchell, ed., *Computers in the Humanities*, Edinburgh Univ. Press.

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.

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.

Coleridge, S.T. (1906). *Biographia Literaria*. Dent, London. [First edition, 1817.]

Dasarathy, B.V. (1991) ed. *Nearest Neighbour (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California.

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.

Dixon, P. & Mannion, D. (1993). Goldsmith's Periodical Essays: a Statistical Analysis of Eleven Doubtful Cases. *Literary & Linguistic Computing*, 8(1), 1-19.

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.]

Eliot, T.S. (1963). *Collected Poems 1909-1962*. Faber & Faber Limited, London.

Elliott, W.E.Y. & Valenza, R.J. (1991). A Touchstone for the Bard. *Computers & the Humanities*, 25, 199-209.

Everitt, B.S. & Dunn, G. (1991). *Applied Multivariate Data Analysis*. Edward Arnold, 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. (1995). *Stylistic Structures: a Computational Approach to Text Classification*. Unpublished Doctoral Thesis, Faculty of Science, University of Nottingham.

Goldberg, J.L. (1995). CDM: an Approach to Learning in Text Categorization. *Proc. 7th IEEE International Conf. on Tools with Artificial Intelligence*.

Greenwood, H.H. (1995). Common Word Frequencies and Authorship in Luke's Gospel and Acts. *Literary & Linguistic Computing*, 10(3), 183-187.

Hamilton, A., Madison, J. & Jay, J. (1992). *The Federalist Papers*. Everyman edition, edited by W.R. Brock: Dent, London. [First edition, 1788.]

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.

Hughes, E.J. (1993). *A Choice of Emily Dickinson's Verse*. Faber & Faber Limited, London.

Johnson, S. (1927). *The History of Rasselas, Prince of Abyssinia*. Clarendon Press, Oxford. [First edition 1759.]

Johnson, T.H. (1970) ed. *Emily Dickinson: Collected Poems*. Faber & Faber Limited, London.

Kamm, A. (1993). *Biographical Dictionary of English Literature*. HarperCollins, Glasgow.

Kjell, B. (1994). Authorship Determination Using Letter Pair Frequency Features with Neural Net Classifiers. *Literary & Linguistic Computing*, 9(2), 119-124.

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.

Makridakis, S. & Wheelwright, S.C. (1989). *Forecasting Methods for Managers*, fifth ed. John Wiley & Sons, 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.

«Feature-Finding for Text Classification»

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.

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.

McMahon, L.E., Cherry, L.L. & Morris, R. (1978). Statistical Text Processing. *Bell System Technical Journal*, 57(6), 2137-2154.

Merriam, T.V.N. (1989). An Experiment with the Federalist Papers. *Computers & the Humanities*, 23, 251-254.

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.

Minitab Inc. (1991). *Minitab Reference Manual, Release 8*. Minitab Inc., Philadelphia.

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.]

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.]

Pound, E.L. (1977). *Selected Poems*. Faber & Faber Limited, London.

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.

«Feature-Finding for Text Classification»

Pudney, J.S. (1976). *For Johhny: Poems of World War II*. Shepheard-Walwyn, London.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California.

Reeves, C.R. (1995). Modern Heuristic Techniques for Combinatorial Problems. Mc-Graw-Hill International, London.

Siedlecki, W. & Sklansky, J. (1989). A Note on Genetic Algorithms for Large-Scale Feature Selection. *Pattern Recognition Letters*, 10, 335-347.

Smith, M.W.A. (1985). An Investigation of Morton's Method to Distinguish Elizabethan Playwrights. *Computers & the Humanities*, 19, 3-21.

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.

Ule, L. (1982). Recent Progress in Computer Methods of Authorship Determination. *ALLC Bulletin*, 10(3), 73-89.

Wickmann, D. (1976). On Disputed Authorship, Statistically. *ALLC Bulletin*, 4(1), 32-41.

Wolff, J.G. (1975). An Algorithm for the Segmentation of an Artificial Language Analogue. *Brit. J. Psychology*, 66(1), 79-90.

Yeats, W.B. (1961). *The Collected Poems of W.B. Yeats*. Macmillan & Co. Limited., London.