Annotated Suffix Trees for Text Modelling and Classification

Rajesh M. Pampapathi

January 2008

A Dissertation Submitted to

Birkbeck College, University of London

in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy

School of Computer Science & Information Systems

Birkbeck College

University of London

Abstract

Suffix trees are compact and versatile data structures in which paths from the root to nodes represent substrings of the encoded text. By annotating such a tree with the frequencies of substrings, it is possible to construct a compact model of text that captures its sequential nature. This thesis investigates the use of such a model in the representation and classification of text.

The basic approach in this thesis is to use an Annotated Suffix Tree (AST) to represent a pre-specified collection of texts ("class"). A document, represented as a string or another ("auxiliary") suffix tree, is matched to the AST to allow, firstly, the scoring of matches between the document and the AST and, secondly, the identification of a number of substrings ("features") that maximally contribute to the matching score. Based on this, methods are proposed for the interrelated problems of: (i) classification of text against several, possibly overlapping, classes, (ii) highlighting the features in a text which are most relevant to a particular class (this problem, to our knowledge, has never before been computationally addressed).

The developed methods are applied to well-established text analysis problems such as e-mail spam filtering and document classification, with three aims in mind: (i) to adjust parameters of the scoring function and assess the effect on performance, (ii) to test the method on benchmark and newly developed test sets, and (iii) to generate human-readable evaluations of classification features within query documents.

Experiments show that the AST method is competitive with other current approaches and in some cases, such as spam filtering, achieves higher classification accuracy; the method also allows the tackling of problems not typically addressed by current alternative methods. The AST method is therefore a useful addition to the arsenal of available classification methods.

Acknowledgments

I would first of all like to thank my supervisors, Boris Mirkin and Mark Levene, for their patient advice, support, and detailed criticism; as well for the freedoms they gave me and for their faith in my work. I am also grateful to Trevor Fenner, for his many sharp insights.

I would like to extend my gratitude to Roger Mitton, Nigel Martin and Trevor Fenner whose lectures were the first I attended in Computer Science and are still among the best I have ever attended in any discipline; they sparked an interest and laid the foundations that have carried me since.

I am grateful to the National Policing Improvement Agency (formerly known as the Police Information Technology Organisation) for funding this work and particularly to Mike Hu for initiating this project. Thanks also to PITO Knowledge Team members Nick Tischler, Tricia Ford, Janina Craske and Kelly Gill for their helpful comments. A special thanks to Freddy Choi, who, though he was not directly connected with the project, gave generously and enthusiastically of his time.

A heartfelt thanks to all the staff, students and researchers at Birkbeck College London and the London Knowledge Lab for the many cool beers and heated conversations which have made my time here such a rich and pleasant one. I would particularly like to thank Lucas Zamboulis and George Papamarkos for the frequent help they gave so freely and the friendship they gave so warmly. I am also grateful to the many others who have helped me in one way or another throughout my PhD, among them are: George Roussos, Boris Galitsky, Kevin Keenoy, Richard Wheeldon, Michael Zoumboulakis, Jenson Taylor, Dionisis Dimakopoulos and Zhu Zheng.

Finally, my warmest thanks go to all my friends for continuing to believe in me even at times when I no longer believed in myself; and above all, to Helge, for all this and more - more than I can say.

To Helge

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

The aim of this thesis is to develop an approach to automated text classification which can address a number of weakness that exist in traditional approaches. By ‘traditional’ I mean methods that are based on the ‘bag-of-words’ vector space model which goes back at least as far as early solutions to problems in Information Retrieval [3, 50]. In such a traditional model, the context in which a word appears is considered irrelevant to the classification task, which is then accomplished purely on the basis of words counts. We attempt to weaken this strong assumption of word independence by taking as features not independent words, but variable length sequences of words, so as to retain something of their context.

Text is a sequence of both words and characters (indeed, it can be seen as a sequence of even larger units such as clauses, sentences and paragraphs) and sequential approaches to text can model text in either way; we attempt in this thesis to develop a uniform approach to both ‘levels’ or ‘term-bases’ of text modelling and analysis. As such, the approach described in this thesis is agnostic as to whether the smallest unit of consideration is a word or a character, and hence we will often refer simply to ‘terms’ as a way of referring to whatever is the smallest unit of text that is being considered. In other words, throughout this thesis we are generally concerned with text seen as a sequence of terms, whatever those terms may be.

We will argue that text should be seen at whatever level is appropriate to the specific task, domain or data set that the classifier is being applied to. Hence, we will see that in domains such as spam email filtering (Chapter 4), the classifier may benefit greatly from a character-based treatment of the text; while on the other hand, when it is applied to a data set such as Reuters (Chapter 5), a word-based treatment may be adequate and/or desirable. That said, we will further argue and show that character-based approaches can compete well with word-based approaches even when applied to data sets such as Reuters, which is normally seen as the bastion of the latter.

To accomplish such modelling of text, we utilise a well know data structure called a suffix tree (or trie) (Chapter 2). The suffix tree is central to this work, and much of the approach we develop is couched in tree representations of the term-sequences that make up text. However, it is certainly possible to represent the term-sequences in text without recourse to suffix trees or, indeed, any other graphical representation - for example, see Cavnar and Trenkle [7], who use a hash-table to represent n-grams (of characters) of variable order (i.e. for varying values of *n*) and their frequencies. Also, see Guthrie '94 [21] for an analysis of n-gram approaches. But framing the problem, and the solution, in the context of suffix trees allows us to present class profiles, query documents, and scoring functions in simple and elegant ways (see Chapters 2 and 3).

Although the suffix tree is central to this work, we do not consider its properties in detail. There is a great deal of good work on suffix trees that provide better exclusive focus for the interested reader. We provide references to recommended material that address many issues regarding suffix tree in Chapters 2 and 3.

Other researchers have also attempted to address the weaknesses in traditional approaches to text modelling and classification, and many, like us, attempt to develop methods which better model the sequential nature of text; we will address these alternative approaches in Chapter 5.

Additionally, this thesis attempts to address a weakness that is not widely addressed in the literature: that classification methods typically offer little in the way of justification for the classification decisions that are made. Hence, the user of a method may obtain a class or set of classes from a classifier, but may be left with little notion of which features in a text led to its allocation to the subset of classes returned by the classifier. Hence, in Chapter 6, we attempt to provide the user with some 'visualisation' of the features that were highly significant in the classification process.

Having developed the method we demonstrate its viability by experimentation on benchmark and in-house (which we have now made publicly available) data sets. Our results are compared against those reported in the literature. Chapter 4 concentrates on anti-spam email filtering as a special case of text classification in which character-level analysis is of particular benefit. Chapter 5 then broadens the application to more general text classification, utilising the Reuters benchmark data set, and analyses both word-level and character-level text models and classifiers. Reuters is chosen as a challenging test for character-based approaches exactly because it is generally considered to be highly amenable to word-based analysis.

We begin, in Chapters 2 and 3, by developing the theoretical underpinnings of the annotated suffix tree approach to text classification. Then in Chapters 4, 5 and 6, we apply the methods to experimental data sets, present the results and compare them to results in the literature. Chapter 7 concludes and describes some possible future directions to this work.

Chapter 2

Annotated Suffix Trees (ASTs)

A suffix tree is a rooted directed tree in which the paths from the root to leaf nodes represent the suffixes of the string it encodes. The data structure is a powerful one which can be used to compactly store a great deal of information regarding the strings it encodes. It has been studied for over 30 years, with linear-time construction algorithms proposed in the ‘70s [58, 37] and again in ‘97 [56]. In recent years, it has been used in fields such as computational biology on applications in DNA sequence matching [4, 34], but its use in natural language modelling is limited. Zamir and Etzioni [61] proposed a clustering algorithm based on the suffix tree and applied it to the organisation of web search results based on the snippets returned by internet search engines; and more recently, Chim and Deng [9] have proposed an alternative clustering algorithm for similar purposes. However, to our knowledge, the suffix tree has never been been utilised more generally in natural language text classification until we proposed it and developed the methods presented in this thesis. Researchers have, on the other hand, suggested the use of substring kernels in support vector machines [33] and suffix trees are a good basis for such an approach; indeed, the application of the methods presented in this thesis to support vector machine kernels would be a very natural branch.

Although the suffix tree is a very powerful data structure, as a concept it is in essence a very simple one. In the context of statistical language processing and classification, it can be seen as simply moving from a bag-of-words model, in which all individual words are independent of each other, to one which maintains information about the context of words using a finite, but variable, memory length.

Such models of text are generally referred to as $n - gram$ models [24]; and suffix trees are an efficient representation for such features [25]. Hence, such an approach is certainly not new, but previous work typically considers $n - grams$ for a particular value of n (that is, for a particular length, n, of characters or words), but the suffix tree allows us to consider $n - grams$ for varying values of n simultaneously and with little extra effort, thus making an $n - gram$ approach more versatile. This effectively makes the suffix tree approach we describe, an ”all-n-gram” approach.

Indeed, our attention in this thesis is focused more on substrings than suffixes and we are thereby able to loosen a little our definition and implementation of suffix trees. A strict definition of a suffix tree, taken from Gusfield 97 [20], would be as follows.

Definition 1. *A suffix tree T of a string s of length m is a rooted directed tree with exactly m leaves labelled 1 to n. Each internal node, other than the root, has two or more children and each edge is labelled with a nonempty substring of s. No two edges out of a node can have edge-labels beginning with the same character. The concatenation of the edge-labels from the root to any node i spells out a suffix of s beginning at the* i^{th} *character.*

To ensure that the number of leaves is indeed n, a terminal character, which appears nowhere else in s, is added to the end of s. Typically, the dollar sign, ’\$’, is used to represent this special character. Without it, there would be the possibility that some suffix of s is equivalent to the prefix of some other suffix of s, which would mean that the path it formed in the tree would not end at a leaf node, but part way along an edge.

In this thesis, we implement suffix trees that differ from the standard one in Definition 1 because we are primarily interested in substrings and their frequencies, rather than, for example, the location of suffixes in the training corpus. Therefore, there is no need for each leaf node to be labelled with a number which provides us the index in s at which the associated suffix begins.

In the next sections we take a closer look at suffix trees as they are implemented and used in this thesis.

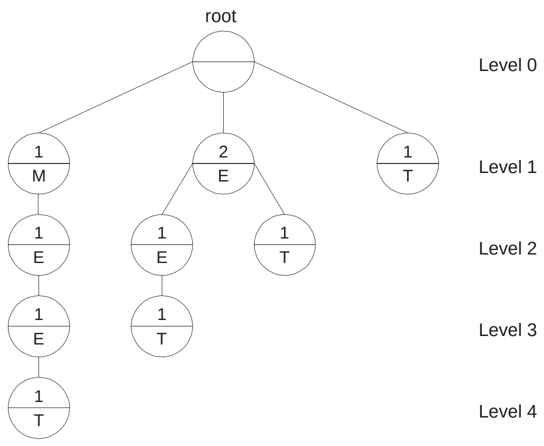


Figure 2.1: A Suffix Tree after the insertion of “meet”.

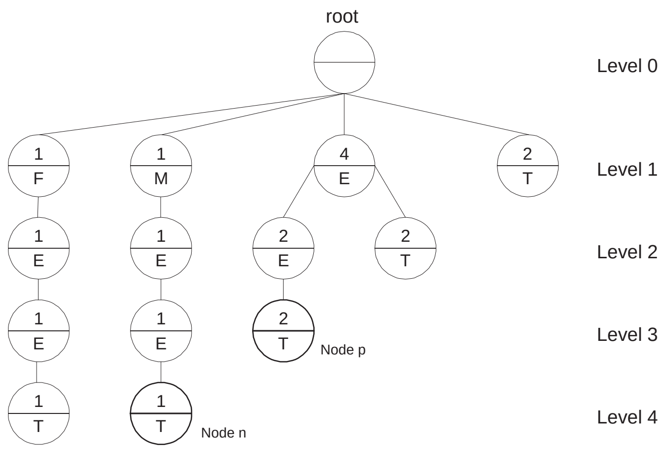


Figure 2.2: A Suffix Tree after insertions of strings “meet” and “feet”.

2.1 Constructing Tree Profiles

We provide a brief introduction to suffix tree construction. For a more detailed treatment, along with algorithms to improve computational efficiency, the reader is directed to Gusfield 97 [20]. As we have said, our representation of a suffix tree differs from the literature, and it does so in two ways that are specific to our task: first, we label nodes and not edges, and second, we do not use a special terminal character. The former has little impact on the theory and allows us to associate frequencies directly with characters and substrings. The later is simply because our interest is actually focused on substrings rather than suffixes; the inclusion of a terminal character would therefore not aid our algorithms, and its absence does not hinder them. Furthermore, our trees are depth limited, and so the inclusion of a terminal character would be meaningless in most situations.

Suppose we want to construct a suffix tree from the string, $s =$ "*meet*". The string has four suffixes: $s(1) =$ "*meet*", $s(2) =$ "*eet*", $s(3) =$ "*et*", and $s(4) =$ "*t*".

We begin at the root of the tree and create a child node for the first character of the suffix $s(1)$. We then move down the tree to the newly created node and create a new child for the next character in the suffix, repeating this process for each of the characters in this suffix. We then take the next suffix, $s(2)$, and, starting at the root, repeat the process as with the previous suffix. At any node, we only create a new child node if none of the existing children represents the character we are concerned with at that point. When we have entered each of the suffixes, the resulting tree looks like that in Figure 1. Each node is labelled with the character it represents and its frequency. The node's position also represents the position of the character in the suffix, such that we can have several nodes labelled with the same character, but each child of each node (including the root) will carry a character label which is unique among its siblings. We refer to the path from the root to the node as the *context* of the node.

If we then enter the string, $t =$ "$feet$", into the tree in Figure 2.1, we obtain the tree in Figure 2.2. The new tree is almost identical in structure to the previous one because the suffixes of the two strings are all the same but for $t(1) =$ "$feet$", and as we said before, we need only create a new node when an appropriate node does not already exist, otherwise, we need only increment the frequency count. We refer to the trees that result from this process as "*Annotated Suffix Trees*" (ASTs) because they essentially resemble structures normally referred to as "*suffix trees*", but carry at each node additional term-sequence frequency annotations.

As we continue to add more strings to this AST, the number of nodes increases only if the new string contains substrings which have not previously been encountered. It follows that given a fixed alphabet and a limit to the length of substrings we consider, there is a limit to the size of the tree. Practically, we would expect that, for any collection of strings, as we add strings from the collection to an AST, the tree will increase in size at a decreasing rate, and will quite likely stabilise. We would conjecture that the size of a stable tree reflects the complexity of the collection of strings it encodes; and we use this intuition in Chapter 4 to devise some normalisation coefficients for use when scoring classes (which we view simply as collections of strings). When a single AST encodes a collection of strings which all belong to one class, we will often refer to it as a *class tree* (see Section 2.2, where we take a closer look at class trees); we also think of a class tree as *profiling* the class it represents, so we will often refer to these as *AST profiles of a class* or *AST class profiles*.

As we have said, in practice, it is not possible to build suffix trees as deep as documents are long and so we limit the depth of the tree to some depth d.

Strictly, the graphs in Figures 2.1 and 2.2 are referred to as a "trie": definitions of a suffix tree require that each internal node has at least two children [20]; and so, any node with just one child is concatenated with that child to form a single node, thereby obtaining the compressed "tree". However, the methods we describe in this thesis are not generally dependent on the specific characteristics of one or the other structure and any description in this thesis that refers to one framework may be reformulated (with some effort) to apply to the other, so we make no distinction here and refer only to "*annotated* suffix trees", in which we allow nodes that have only one child. We look at the particular characteristics of ASTs that are relevant to us in the next section.

2.2 Class Trees

For any string s let us designate the i^{th} character of s by s_i; the suffix of s beginning at the i^{th} character by $s(i)$; and the substring from the i^{th} to the j^{th} character inclusively by $s(i, j)$.

Any node, n, labelled with a character, c, is uniquely identified by the path from the *root* to n. For example, consider the tree in Figure 2.2. There are several nodes labelled with a "t", but we can distinguish between node $n = (\text{“}t\text{” } given \text{ “}mee\text{”}) = (t|mee)$ and $p = (\text{“}t\text{” } given \text{ “}ee\text{”}) = (t|ee)$; these nodes are labelled n and p in Figure 2.2. We say that the path of n is $\overrightarrow{P}_n = \text{“}mee\text{”}$, and the path of p is $\overrightarrow{P}_p = \text{“}ee\text{”}$; furthermore, the frequency of n is 1, whereas the frequency of p is 2; and saying n has a frequency of 1, is equivalent to saying the frequency of "t" given "mee" is 1, and similarly for p.

If we say that the *root* node, r, is at level zero in the tree, then all the children of r are at level one. More generally, we can say that the level of any node in the tree is one plus the number of letters in its path. For example, $level(n) = 4$ and $level(p) = 3$. Levels correspond to depth: so a tree of maximum depth 1 consists only of level 1 nodes.

The set of letters forming the first level of a tree is the *alphabet*, Σ - meaning that all the nodes of the tree are labelled with one of these letters. For example, considering again the tree in Figure 2.2, its first level letters are the set, $\Sigma = \{m, e, t, f\}$, and all the nodes of the tree are labelled by one of these.

Suppose we consider a class, C, containing two strings (which we might consider as documents), $s =$ "*meet*" and $t =$ "$feet$". Then we can refer to the tree in Figure 2.2 as the *class tree* of C, or the *AST profile* of C; which we denote by T_C.

The size of the tree, $|T_C|$, is the number of nodes it has, not counting the root node, and it has as many nodes as C has unique substrings. For instance, in the case of the tree in Figure 2.2:

$$UC = \texttt{uniqueSubstrings}(C) = \left\{ \begin{array}{l} meet, mee, me, m, eet, ee, e, et, t, \\ feet, fee, fe, f \end{array} \right\}$$

$$|UC| = |\texttt{uniqueSubstrings}(C)| = 13$$

$$|T_C| = \texttt{numberOfNodes}(T_C) = 13$$

This is clearly not the same as the total number of substrings in C:

$$AC = \texttt{allSubstrings}(C) = \left\{ \begin{array}{l} meet, mee, me, m, eet, ee, e, et, e, t, \\ feet, fee, fe, f, eet, ee, e, et, e, t \end{array} \right\}$$

$$|AC| = |\texttt{AllSubstrings}(C)| = 20$$

As an example, note that the four "e"s in the set are in fact the substrings s(1,1), s(2,2), t(1,1) and t(2,2).

Furthermore, as each node in the tree, T_C, represents one of the substrings in UC, the size of the class, AC, is equal to the sum of the frequencies of nodes in the tree T_C.

$$|AC| = |\texttt{allSubstrings}(C)| = \texttt{sumOfFrequencies}(T_C) = 20$$

In a similar way, the annotated suffix tree allows us to read off other frequencies very quickly and easily. For example, if we want to know the number of characters in the class C, we can sum the frequencies of the nodes on the first level of the tree; and if we want to know the number of substrings of length 2, we can sum the frequencies of the level two nodes; and so on.

This also allows us to very easily estimate probabilities of substrings of any length (up to the depth of the tree), or of any nodes in the tree. For example, we can say from the tree in Figure 2.2, that the probability of a substring, u, of length two, having the value, $u =$ "*ee*", given the class C, is the frequency, f, of the node $n = (e|e)$, divided by the sum of the frequencies of all the level two nodes in the tree T_C:

$$\texttt{estimatedLevelProbability}(u) = Pr(u|l_u) = \frac{f(u)}{\sum_{i \in L_u} f(i)} \tag{2.1}$$

where l_u is the level of u; and L_u is the set of all nodes at the same level as u.

Similarly one can estimate the conditional probability of u as the frequency of u divided by the sum of the frequencies of all the children of u's parent:

$$\texttt{estimatedConditionalProbability}(u) = Pr(u|\overrightarrow{P}_u) = \frac{f(u)}{\sum_{i \in n_u} f(i)} \tag{2.2}$$

where n_u is the set of all children of u's parent. Or the probability of the particular substring represented by u against all the possible substrings given the tree:

$$\texttt{estimatedTotalProbability}(u) = Pr(u) = \frac{f(u)}{\sum_{i \in T} f(i)} \tag{2.3}$$

The sum, $\sum_{i \in n_u} f(i)$, in the denominator of Equation 2.2 is equal to the frequency of u's parent minus the number of terminations that occurred at u's parent. As our trees are depth limited to a depth, d, which is typically small compared to the size of documents, such termination will be rare and the sum over sibling frequencies will generally be approximated well by the frequency of the parent of the siblings. It is arguable that taking the frequency of the parent as the denominator value for Equation 2.2 would give a better indication of the probability of child nodes, because terminations should be included as part of the sample space. However, for practical purposes, the parent frequency will be approximately equal to the sum of the sibling frequencies and so we use this approximation from now on - hence, we assume that:

$$\sum_{i \in n_u} f(i) \equiv frequency(parent(u)) = f(\Upsilon_u) \tag{2.4}$$

where Υ_u indicates the parent of u.

2.3 Multi-Class Trees

We can tackle the multi-class case in two ways. Either we construct a separate class tree for each class or we construct one suffix tree which represents all the encountered character sequences and annotate each node with a set of classes indicating in which classes the particular substring/node has been encountered and a corresponding set of frequencies indicating how often the node was encountered in each class.

In such a scenario, we would now distinguish between the frequency of a node given a class, $f(u|c)$; and the frequency of the node independent of class, $f(u)$. If the context is not clear, we will specify the tree from which frequencies are being read - e.g. $f(u|T,c)$ or $f(u|T_c)$, and $f(u|T)$, respectively. If for a certain class, c_i, the substring represented by node u has not occurred, then $f(u|c_i)$, the class dependant frequency for u given c_i, would be zero. The class independent frequency of u would be the sum of all the class dependent frequencies of u: $f(u) = \sum_{c \in C} f(u|c)$.

2.3.1 Class Frequencies

For any given class, c, we can think of at least three different frequency distributions over all the strings encountered and represented in the suffix tree:

- Class dependent frequency: $f(u|c)$
- Class independent frequency: $f(u) = \sum_{c \in C} f(u|c)$
- Negative class Frequency: $f(u|\overline{c}) = f(u) - f(u|c)$

2.3.2 Class Probabilities

From these three frequency distributions, we can derive corresponding probability distributions. We can do this with conditional probabilities, level probabilities, or with total probabilities as we did in Section 2.2. However, for the sake of brevity, let us consider only the conditional probabilities; corresponding *level* and *total* probabilities may also be defined.

The class dependent conditional (i.e. path dependent) probability (CDCP) is given by:

$$\begin{aligned} \texttt{CDCP}(u,c) &= Pr(u|\overrightarrow{P}_u, c) = \frac{f(u|c)}{\sum_{n_i \in N} f(n_i|c)} \\ &= \frac{f(u|c)}{f(\Upsilon_u|c)} \end{aligned} \tag{2.5}$$

; where N is the set of nodes which are the children of u's parent - i.e. u and its siblings.

The class independent conditional (path dependent) probability (CINDCP) is given by:

$$\begin{aligned} \texttt{CINDCP}(u) &= Pr(u|\overrightarrow{P}_u) = \frac{f(u)}{\sum_{n_i \in N} \sum_{c_i \in C} f(n_i|c_i)} \\ &= \frac{f(u)}{\sum_{n_i \in N} f(n_i)} \\ &= \frac{f(u)}{f(\Upsilon_u)} \end{aligned} \tag{2.6}$$

; where, again, N is the set of nodes which are the children of u's parent.

And finally, the negative class dependent conditional probability (NCDCP) is given by:

$$\begin{aligned} \texttt{NCDCP}(u,\overline{c}) &= Pr(u|\overrightarrow{P}_u, \overline{c}) = \frac{f(u) - f(u|c)}{\sum_{n_i \in N} f(n_i)} \\ &= \frac{f(u) - f(u|c)}{f(\Upsilon_u)} \end{aligned} \tag{2.7}$$

; where, again, N is the set of nodes which are the children of u's parent.

Throughout this thesis, we mainly make use of these three conditional probabilities. However, we use them primarily as weights indicating the significance of the feature represented by the node, rather than as strict probabilities.

Chapter 3

Classification using ASTs

Researchers have tackled the problem of the construction of a text classifier in a variety of different ways, but it is popular to approach the problem as one that consists of two parts:

1. The definition of a function, $CS_i : D \rightarrow \mathbb{R}$, where D is the set of all documents; such that, given a particular document, d, the function returns a *category score* for the class i. The score is often normalised to ensure that it falls in the region $[0,1]$, but this is not strictly necessary, particularly if one intends, as we do, simply to take as the class prediction the highest scoring class (see Section 3.2 below). The interpretation of the meaning of the function, CS, depends on the approach adopted. For example, in naive Bayes (Section 4.2), $CS(d)$, is interpreted as a *probability*; whereas in other approaches such as Rocchio [48], $CS(d)$ is interpreted as a *distance* or *similarity* measure between two vectors.

2. A decision mechanism (or rule) which determines a class prediction from a set of class scores. For example, the highest scoring class might be taken as the predicted class: $PC = argmax_{c_j \in C}\{CS_j(d)\}$. Alternatively, if $CS(d)$ is interpreted as a value with definite range, such as a probability, the decision may be based on a threshold, th, such that the predicted class is taken as c_j if $CS_j(d) > th$, and as not c_j otherwise.

Lewis et al. [31] and Duda and Hart [14] refer to probabilistic models such as naive Bayes as parametric classifiers because they attempt to use the training data to estimate the parameters of a probability distribution, and assume that the estimated distribution is correct. Non-parametric, geometric models, such as Rocchio [48], instead attempt to produce a profile or summary of the training data and use this profile to query new documents to decide their class.

It is possible to approach the construction of a suffix tree classifier in either of these two ways and indeed a probability-based approach has been developed by Bejerano and Gill [4] for use in gene sequence matching. However, Bejerano and Gill [4] did not find the suffix tree entirely convenient for developing a probabilistic framework and instead developed a probabilistic analogue to the suffix tree and used this modified data structure to develop probabilistic matching algorithms.

In this thesis, the original structure of the suffix tree is retained and we focus on the development of a *geometric* approach, in which a match between a document and an annotated suffix tree profile of a class is a set of coinciding substrings each of which must be scored individually so that the total score is the sum of individual scores. This is analogous to the inner product between a document vector and class profile vector in the Rocchio algorithm [48].

We did experiment with probabilistic models, and found that it was possible to construct one without altering the structure of the suffix tree, but found performance was better using scoring functions which simply approximated probabilistic models (see Section 3.1).

With regard to the second issue in classifier construction – that of a decision mechanism – we develop different decisions mechanisms for each approach and present them in Section 3.2. First, we develop an approach to scoring.

3.1 Scoring: The Overlap Mass Score

We found that an additive scoring algorithm, which we now refer to as the *overlap mass score* (OMS) performed better than others we experimented with. The method is governed by three heuristics:

H1 Each substring $s(i)$ that a string s has in common with a class T indicates a degree of similarity between s and T, and the longer the common substrings the greater the similarity they indicate.

H2 The more diverse[1] a class T, the less significant is the existence of a particular common substring $s(i, j)$ between a string s and the class T .

[1]*Diversity* is here an intuitive notion which the scoring method attempts to define and represent in a number of different ways.

H3 The more diversely a string, m, occurs across all classes, the less significant is the existence of a particular common substring $s(i,j) = m$ between a string s and the class T .

We present two forms of the overlap mass scoring (OMS) algorithm based on these three heuristics. The first treats queries as strings and the second treats them as suffix trees. In essence these two approaches are analogues of each other and generally have equivalent evaluations. However it is useful to present the query-as-string method first because it is initially a more natural way in which to approach the problem and is then easily extended to the query-as-tree model.

3.1.1 Scoring Queries as Strings

Treating a query as a string is quite natural. The query may be a set of key-words, a phrase or a document. In the context of a classification task, the query is typically a document, the class of which we wish to establish. The document is taken as a single string, s, which is itself a sequence of terms (be they words or characters):

$s = s_1 s_2 s_3 ... s_N$.

From the string s we may obtain a set of its suffixes:

$suffixes(s) = \{s(1), s(2), s(3) ... s(N)\}$

As our tree will be depth limited to some depth d, considering each suffix is practically equivalent to considering a set of substrings:

$substrings(s,d) = \{s(1,d+1), s(2,d+2) ... s(N-d,N), s(N-d-1,N) ... s(N,N)\}$

For each suffix (or substring of length d) $s(i)$ we may identify a match m_i between it and the AST profile tree, where a match is defined as the longest prefix of $s(i)$ which corresponds to a rooted path in the profile tree. Hence, from $s(i)$, we may obtain the match, m_i:

$m_i = s(i,j) = m_1 m_2 m_3 ... m_j = s_i s_{i+1} s_{i+2} ... s_j$

such that $j - i < d$.

Thus, for any query string, s, there will be a possibly empty set of matches, M, between it and the profile tree, T, which represents an *overlap* between s and T. Each item, $m_i \in M$, is a sequence of terms that occurs in both the profile tree (as a path from the root) and the query string (as a substring). Furthermore, if $d \leq |s|$, then it is possible that for any two matches, $m_x \in M$ and $m_y \in M$, that $m_x = m_y$. In fact, as d is typically small compared to $|s|$, it will generally be possible to group the matches in M

into **equivalence sets** in which all matches are constituted by the same sequence of terms although each sequence in such an equivalence set will be derived from a different location in the query string. We take advantage of this fact when we develop further the overlap scoring method in Section 3.1.2.

The process of scoring the query string, s, boils down to the evaluation of the set M, which itself boils down to the evaluation of its members, $m \in M$, which in turn boils down to the evaluation of each of the terms in the sequence that constitutes m. We refer to these *levels* of evaluation as *document-level*, *match-level* and *term-level*, respectively.

In the next three sections we present the OMS approach to scoring at each of these levels (document, match and term) before we consider them together and elaborate on the underlying motivation for the scoring framework. Section 3.1.2 then presents an alternative way of describing the overlap set M and the OMS method.

Conceptually dividing the scoring into these three levels allows us to introduce and experiment with levels of normalisation: *term-level*, reflecting information about single terms, *match-level*, reflecting information about strings; and *tree-level*, reflecting information about the class as a whole. We experimented with normalisation at each of these three level and found each of them to be useful under certain conditions, as we will see in Chapter 4. However, the OMS method itself does not hinge on any such normalisation.

Throughout this thesis, we will use two different *term-bases*: *characters* and *words*. Hence, the term-level will often be referred to as *character-level* or *word-level* depending on whether our analysis is done on a *character-base* or a *word-base*. Similarly, depending on the *term-base*, the classifier we construct will often be referred to as a *character-based* or *word-based* classifier.

Scoring a document

In practice the scoring is done by evaluating each suffix of the string in turn to discover the longest match between it and the profile tree. As we iterate through the string, we sum over all the individual match scores and hence the total overlap mass score (OMS) for a query string (or document) will be the sum of the individual match scores (MS):

$$OMS(s,T) = \sum_{i=0}^{n} MS(s(i),T) \tag{3.1}$$

; where the function, $MS(s(i),T)$, searches for a match, m, between suffix $s(i)$ and tree T, and if one is found, scores it according to Equation 3.2 below.

We experimented with a number of approaches to tree-level normalisation of the sum in Equation 3.1, motivated by heuristic H2 and based on tree properties such as *size*, as a direct reflection of the diversity of the class; *density* (defined as the average number of children over all internal nodes), as an implicit reflection of the diversity; and total and average *frequencies* of nodes as an indication of the size of the class[2]; but found none of our attempts to be generally helpful to the performance of the classifier.

Scoring a match

The scoring of a match is itself nothing more than the normalised sum of the scores for each term. Hence, the score, $MS(m)$, for a match $m = m_0 m_1 m_2 ... m_n$, has two parts, firstly, the scoring of each term (and thereby, each substring), m_i, with respect to its conditional probability, using a *significance* function of probability, $\phi[\hat{p}]$, defined below, and secondly, the adjustment (normalisation), $\nu(m|T)$, of the score for the whole match with respect to its probability in the tree:

$$MS(m,T) = \nu(m|T) \sum_{i=0}^{n} \phi[\hat{p}(m_i|T)] \tag{3.2}$$

We experimentally investigate definitions of match-level normalisation in Chapter 4.

The Significance Function

The key to the OMS method is the *significance function*. How we define it can have a strong effect on the behaviour and performance of the classifier. The simplest definition actually ignores the probability of the character altogether; we call this a *constant* significance function:

$$\phi[p(m_i)] = 1 \tag{3.3}$$

If the scoring method were used in this form to score the similarity between two strings, it would simply count the number of substrings that the two strings have in common. For example, suppose we

[2]Class size is defined as the total number of substrings in the documents of the class, and tree size as the number of nodes in the tree, that is, the number of unique substrings in the class (see Section 2.1).

have a string $t =$ "*abcd*". If we were to apply this scoring function to assessing the similarity that t has with itself, we would obtain a result of 11, because this is the number of unique substrings that exist in t. If we then score the similarity between t and $t^0 =$ "$Xbcd$", we obtain a score of 6, because the two strings share 6 unique substrings; similarly, a string $t^1 =$ "$aXcd$" would score 4.

Another way of viewing this is to think of each substring of t as representing a feature in the class that t represents. The scoring method then weights each of these as 1 if they are present in a query string and 0 otherwise, in a way that is analogous to the simplest form of weighting in algorithms such as Rocchio [48].

Once seen in this way, alternative definitions of the significance function may be viewed as different approaches to deciding how significant each common substring is, or in other words, deciding how to weight each class feature – in much the same as with other *non-parametric* classifier algorithms.

Hence, the significance function addresses the issue of how to evaluate the importance of a term as it appears in a certain position in the match. Moving beyond the binary weighting of Equation 3.3, an obvious weighting to employ would be the probability (as defined in one of the ways described in Section 2.3.2) of the terms; that is, we define $\phi[p(m_i)]$ as follows:

$$\phi[p(m_i)] = p(m_i) \tag{3.4}$$

where $p(m_i)$ is the conditional probability of m_i.

Using the conditional probability rather than the total or level probability has the benefit of supporting heuristic H1: as we go deeper down the tree, each node will tend to have fewer children and so the conditional probability will be likely to increase; conversely, there will generally be an increasing number of nodes at each level and so the total probability of a particular node will decrease. Indeed we did experiment with the total probability and found that performance was significantly decreased.

Furthermore, by using the conditional probability we also only consider the independent parts of features when deriving scores. So for example, if $m =$ "*abc*", by the time we are scoring the feature represented by "*abc*", we have already scored the feature "*ab*", so we need only score "*c*" given "*ab*".

But we need not adhere to such a *linear* function of the probability. A non-linear approach can help make the classifier more sensitive to specific probability ranges, which may be useful in certain domains - as we will show in Chapter 4.

As a final example, we may also choose a significance function which approximates log-likelihood approaches by specifying:

$$\phi[p(m_i)] = log[p(m_i)] \tag{3.5}$$

but this leaves unaddressed the problem of *null* probabilities – i.e. what to do with terms or term sequences that have not been encountered during training and thereby have a frequency of zero in the data generated model. Typically, the problem is addressed by the use of smoothing methods [62]. For example, Bejerano and Gill [4] address the problem by redistributing the frequencies at each node in the suffix tree across the whole alphabet, thereby effectively growing the suffix tree so that all possible outcomes (i.e. the alphabet of possible outcomes) has an associated probability at each node.

For efficiency reasons, the OMS method aims to only evaluates the overlap between the query and the profile, thereby ignoring any term or term sequences that have not been previously encountered and are therefore not recorded in the profile tree; however, with an appropriate definition of the significance function, the OMS method can still adequately address the problem of nulls. We explore this issue in the next section, which develops the OMS method further by using an AST representation of queries.

The AST model as so far described – using a string representation of queries – was developed and published in two papers [43, 42]. We present experimental results using this framework in Chapter 4.

3.1.2 Scoring Queries as Suffix Trees

We now develop the method by first taking the query document, s, and constructing from it an auxiliary AST, which we call a query tree, Q. We then identify rooted paths in the query tree which have corresponding rooted paths in the profile tree, T. This collection of corresponding nodes defines an *overlap* suffix tree, R; and the paths from the root of R to its leaf nodes will trace character sequences which correspond with the matches in the set M from Section 3.1.1.

As an illustration of the overlap tree R between a query tree, Q, and profile tree, T, consider Figure 3.1, which shows the query tree for the query string, "$feel$", with those nodes that overlap with the profile class tree in Figure 2.2 shown as square nodes and in bold. Figure 3.2 shows the profile tree again with those nodes that overlap with Q's nodes now shown as square.

Viewed in this way, the scoring of a query document against all the class profiles is transformed into

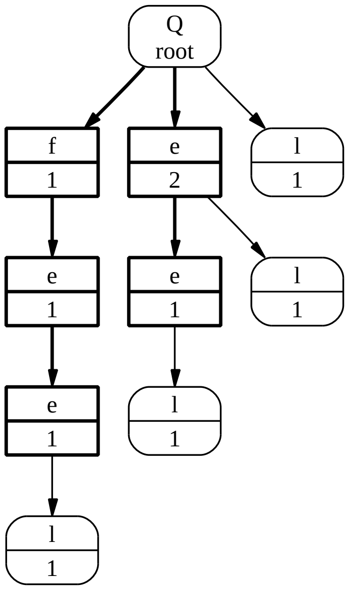


Figure 3.1: A query tree for the query string "*feel*". Those nodes/paths which *overlap* with the earlier class profile are shown in a square shape and in bold.

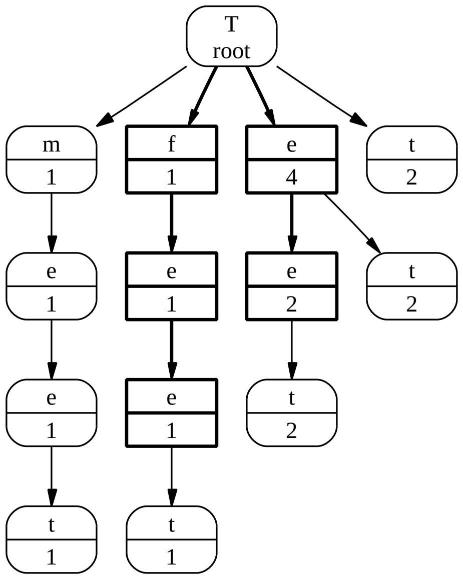


Figure 3.2: A Suffix Tree after insertion of strings "*meet*" and "*feet*".

assessing the similarity between two trees, or in other words evaluating the overlap tree R with respect to each class.

Let us consider R in relation to the query tree, Q, and the the profile tree, T. Then R is structurally similar to a sub-tree of Q, which we denote as Q^+, and to a sub-tree of T, which we denote as T^+. The frequency indicated by nodes in Q^+ give us the number of times the sequence of terms mapped out by the path from R's root to a node, $r \in R$, has been encountered in the query string. In other words, it represents the size of the match *equivalence sets* (see Section 3.1.1) in M. Two matches from the same equivalence set are scored identically using the frequencies or probabilities indicated by the nodes in T^+, so we can in fact calculate the score associated with a node in the overlap tree once and multiply by the size of the equivalence set, which is given by the corresponding node in the overlap sub-tree, Q^+.

It is helpful at this point to introduce some notation so we can say all this more formally. We do this next, before considering the form of the overlap mass function as a tree similarity measure.

Notation

We define a tree, T, as a set of nodes, $t \in T$, each of which is uniquely associated with a path, $\overrightarrow{P}(t)$, from the root of the tree to t. In our case, the tree T is a suffix tree, and so with each node is associated a term, $\omega(t)$, and a frequency $f(t)$; the path $\overrightarrow{P}(t)$ is therefore a sequence of such terms. We define the weight, W, of a tree as the sum of the frequencies of all the nodes in the tree: $W(T) = \sum_{t \in T} f(t)$.

Now consider two trees, T and Q, such that T is a class profile tree and Q a 'query' constructed from a document we wish to classify in one of a set of classes, C.

- For any two trees such as T and Q there exists an 'overlap' set, V_{QT}:
  $V_{QT} = \{(q,t) : q \in Q, t \in T, \overrightarrow{P}(q) = \overrightarrow{P}(t)\}$.
  - Note that $V_{QT} \neq Q \cap T$ because we want to maintain a distinction between nodes which belong to Q and nodes which belong to T so that we can distinguish between their independent frequencies, $f(t)$ and $f(q)$. We will drop the sub-script, 'QT', when the context is clear.
- We then have a set of nodes from Q which are represented in the set V_{QT}:
  $Q^+ = \{q : q \in Q, t \in T, (q,t) \in V\}$

- And we have a set of nodes in Q which are not represented in the set V_{QT}:
  $Q^- = Q/Q^+$
  - Note that Q^- is the relative complement of Q wrt T.
  - Where there is potential confusion, we specify the relative set as a superscript: Q^{-T}.
- Similar, symmetric sets exist for T.

We have defined a set of pairs, V_{QT}, so that we can understand the difference between $f(t)$, as the frequency of a node in T; and $f(q)$, the frequency of a node in Q. However, we obtain a similar result if we use the definition of the overlap sub-tree, Q^+, and distinguish between the frequency of $q \in Q^+$ given the profile tree T, $f(q|\overrightarrow{P}_q, T)$; and the frequency of $q \in Q^+$ given the query tree, Q, $f(q|\overrightarrow{P}_q, Q)$. Under this scheme, for any node, $q \in Q^-$, we can say that $f(q|\overrightarrow{P}_q, T) = 0$. Finally, if we ignore frequency counts for a moment, we can say:

- $Q^+ = T^+ = R$
- $Q^- = Q/Q^+ = Q/R$
- And, $T^+ = T/T^- = T/R$

Tree Similarity Version of OMS

We are now in a position to give expression to the overlap mass score (OMS) as a measure of the significance of the overlap between two suffix trees - or in other words, a measure of the *similarity* between two trees. The evaluation is done over the set V_{QT} such that the for each pair, $(q,t) \in V_{QT}$, such that $f(q)$ tells us how many times the sequence (match) represented by q has occurred and $\phi(t)$ tells us the significance of the sequence (or match). Hence, we can evaluate the OMS between a query Q and profile tree T as:

$$OMS(Q,T) = \sum_{(q,\,t) \in V_{QT}} f(q)\nu(t)\phi(t) \tag{3.6}$$

where $\phi(t)$ is the a significance function on the node $t \in T$ and $f(q)$ is the frequency of node $q \in Q$. We might also express this using our alternative notation:

$$OMS(Q,T) = \sum_{q \in Q^+} f(q|Q)\nu(q|T)\phi(q|T) \tag{3.7}$$

These formulas are not a direct copy of the double sum approach presented in the previous section. This is because a match $m \in M$ is represented in the overlap tree R as a path from the root to a leaf node. However, in Equations 3.6 and 3.7, the match normalisation term is applied to each node, whether it is internal or external.

Notice also, that we have broadened the definition of the *significance function* so that it is now a function of t directly, rather than a function of the probability of t. Doing this results in more generality by giving greater flexibility to our definition of ϕ.

This doesn't mean that we need change the basic definitions of ϕ, and indeed, as we will see in later chapters, some simple definitions work very well. But this more general formulation of the significance function does allow us to experiments with, for example, smoothed probabilistic scoring functions.

In the following sections we will use only the notational representation of Equation 3.7.

Smoothed Probabilistic Scoring Functions

As we have said, probabilistic models suffer from the problem of null probabilities. These arise when a term is encountered in the query document which has not been encountered during training (i.e. during the construction of the model). The probability of such an '*unseen*' term would, under the model, be *zero*, and thereby make the document impossible to evaluate. To overcome this problem it is common to use some sort of smoothing method that specially deals with terms in the query that are previously unseen. Two common approaches to smoothing are, (a) to redistribute the probability mass over all possible outcomes, not only those encountered; and (b) to use two distributions, one for seen terms and one for unseen terms. Laplace smoothing – which we take a closer look at below – and the previously mentioned approach of Bejerano and Gill [4], which we do not describe here, fall into group (a). Approaches falling into group (b) are generally referred to as *mixture models* [35]. However, the two approaches are highly related, as we will see below.

Laplace Smoothing

Laplace smoothing simply adds a count to each possible outcome, so that outcomes which originally have a count of *zero*, then have a count of *one*. This also increases the total probability mass by the size of the sample space. Hence, given a dictionary containing M terms, the probability of a term, d_i, which previously has a frequency given by $f(d_i)$, given the class, c, would under Laplace smoothing be assigned a probability given by:

$$Pr(d_i|c) = \frac{1 + f(d_i|c)}{M + \sum_{k=1}^{M} f(d_k|c)} \tag{3.8}$$

where $f(d_k|c)$ is the frequency, given the class c, of the k^{th} term in the dictionary.

We will see the smoothing shown in Equation 3.8 in Chapter 4 when we compare the suffix tree classification performance to naive Bayes classification. Translating Equation 3.8 into a suffix tree framework, we get:

$$Pr(q|Q,T_c) = \frac{1 + f(q|\overrightarrow{P}_q, T_c)}{|\Sigma| + f(\Upsilon_q|T_c)} \tag{3.9}$$

Using such a smoothed probability of a node along with the (obviously false) assumption that all nodes are independent, we can calculate the log likelihood of a query as:

$$log[Pr(Q|T_c)] = \sum_{Q} f(q|Q) log \left\{ \frac{1 + f(q|\overrightarrow{P}_q, T_c)}{|\Sigma| + f(\Upsilon_q|T_c)} \right\} \tag{3.10}$$

We can then separate Equation 3.10 into two summations: one that evaluates the nodes in Q which belong to Q^+ and nodes which belong to Q^-; noting that for nodes, $q \in Q^-$, $f(q|\overrightarrow{P}_q, T_c) = 0$, we obtain:

$$log[Pr(Q|T_c)] = \sum_{Q+} f(q|Q) log \left\{ \frac{1 + f(q|T_c)}{|\Sigma| + f(\Upsilon_q|T_c)} \right\} + \sum_{Q^-} f(q|Q) log \left\{ \frac{1}{|\Sigma| + f(\Upsilon_q|T_c)} \right\} \tag{3.11}$$

The first summation on the right hand side of Equation 3.11 is equivalent to the overlap mass score with the significance function defined to be the *log* term. The second summation is the smoothing function which addresses any null probabilities.

Equation 3.11 is however unsatisfactory as a probabilistic scoring function because the parent of a node $q \in Q^-$, may also be a member of Q^- - that is to say: $\Upsilon_q \in Q^-$; in which case, we have: $f(\Upsilon_q|T,c) = 0$. But then the *log* term of the second summation will reduce to $1/\Sigma$; which means that the conditional probabilistic score associated with the children of an unmatched term may be higher than the probabilistic score with some matched terms and will certainly be higher than the score associated with the the first unmatched term. Such a result runs counter to Heuristic H1 in that we would like to have an increasing penalty for longer unmatched sequences rather than a decreasing penalty.

Mixture Model Smoothing

An alternative smoothing method is to use two different distributions – i.e. to use a mixture model – one for matched terms and one for unmatched terms. For example, we may assume that the distribution of terms we observe in a query is generated by the *class model* on the one hand and the *corpus model* on the other. For matched terms we use the class model and for unmatched terms we use the corpus model:

$$Pr(Q|T_c) = \sum_{Q+} f(q|Q) log \left\{ \frac{f(q|T_c)}{f(\Upsilon_q|T_c)} \right\} + \sum_{Q^-} f(q|Q) log \left\{ \frac{f(q|T)}{f(\Upsilon_q|T)} \right\} \tag{3.12}$$

where T represents the merged profile trees of all classes and hence $f(q|T)$ is the sum of the frequencies of q over all classes.

If we add and subtract from Equation 3.12 the sum over Q^+ of the logged corpus probabilities:

$$\sum_{Q^+} f(q|Q) log \left\{ \frac{f(q|T)}{f(\Upsilon_q|T)} \right\} \tag{3.13}$$

and rearrange, we obtain:

$$log[Pr(Q|T_c)] = \sum_{Q^+} f(q|Q) log \left\{ \frac{f(q|T_c)}{f(\Upsilon_q|T_c)} \frac{f(\Upsilon_q|T)}{f(q|T)} \right\} + \sum_{Q} f(q|Q) log \left\{ \frac{f(q|T)}{f(\Upsilon_q|T)} \right\} \tag{3.14}$$

As the the second of these summations is the same for all classes and we want nothing more than a probabilistic ranking of the classes, we need only evaluate the first of these summations. Hence, if we define the significance function of the overlap mass score as:

$$\phi(q|Q,T) = log\left\{\frac{f(q|T_c)}{f(\Upsilon_q|T_c)}\frac{f(\Upsilon_q|T)}{f(q|T)}\right\} \tag{3.15}$$

We can obtain an OMS ranking of the classes that is similar to other smoothed probabilistic methods used widely in the language modelling literature. OMS defined in this way cannot be considered as a true probability because the nodes in the suffix tree are not independent, but the significance function in Equation 3.15 has the advantage that it balances the class frequencies using the corpus frequencies so that nodes (and the term sequences they represent) with frequencies above the average frequency of the corpus are given greater significance, while retaining the efficiency of evaluation over only the overlap tree between Q and T.

We experiment with Equation 3.15 in Chapter 5. For a more detailed analysis, the reader is directed to Zhai and Lafferty 2001 [62], who show how such probabilistic models are analogous to heuristic scoring methods such as TF-IDF, which is a popular method derived from the information retrieval literature [3].

Despite the advantages of this ‘probabilistic’ scoring function, we often found, as we will show in later chapters, that far simpler significance functions such as the *root significance function*:

$$\phi(t) = \sqrt{Pr(t|\overrightarrow{P}_t, c)} \tag{3.16}$$

can perform extremely well in certain domains.

Taking the root of the probability makes the classifier more sensitive to variations in lower probabilities which allows it to discern classes with sparse feature distributions, which is often the case in natural language classification tasks.

3.2 Decision Mechanisms

Scoring a query against all the classes in the set of classes, C, gives us a ranking (by score) of the classes. If we know that the query document belongs in only one of these classes, the decision may be made in the relatively straight forward way of simply assigning to the class which ranks the highest. However, we may also want to bias the decision mechanism – to allow fine tuning, for example.

In a two class case, such as that investigated in Chapter 4, we may take a ratio of the two scores;

classifying the query document into one class if it is above a threshold, λ, and in the other class if it is below:

$$\frac{OMS(Q,T_{c_1})}{OMS(Q,T_{c_2})}\begin{cases} > & \lambda & \mapsto c_1 \\ < & \lambda & \mapsto c_2 \end{cases} \tag{3.17}$$

which is equivalent to the Bayesian decision rule commonly used in machine learning [14]. With more classes, we may employ a threshold for each class, whereby a score above the class specific threshold would place the query document in the class, thus allowing multiple class assignment.

$$OMS(Q,T_{C_i}) > \lambda_{c_i} \mapsto c_i : c_i \in C \tag{3.18}$$

Or we may even employ a single threshold for all classes.

Another common method is to look for a decision boundary between the query belonging to the particular class in question or not belonging to it. We ask the same question for each class. Hence:

$$\frac{OMS(Q,T_{c_i})}{OMS(Q,T_{\bar{c_i}})} > \lambda_{c_i} \mapsto c_i \tag{3.19}$$

which is equivalent to Equation 3.17 in the two class case.

Such thresholds are commonly used in machine learning and text classification. However, we additionally develop an alternative method whereby we take the ordered scores for each class and cluster the classes, by score, into a positive and negative cluster. All those classes in the positive cluster are suggested classes for the query document and all those in the negative cluster are not. This method is explained in more detail and investigated in Chapter 5.

3.3 Evaluating Classification Features

Much of the research effort on text analysis and classification is focused on accuracy and performance. Much less effort is focused on providing a potential user with some degree of provenance or justification for the classification decisions made. We attempt to address this gap using a method which highlights the key terms and term sequences in a document which contributed most to the classification decision. We refer to this method as *Document-to-Class Highlighting*.

Search engines such as `Google` offer some ‘justification’ for the documents they return by highlighting the query terms as they occur in the returned documents, but such offerings are limited.

Our overall approach is to take a document and compare it to a class, which we will often refer to as the *relative class*, and highlight those terms and term sequences in the document that are most significant to that class as indicated by its AST class profile.

Because we wish to obtain an evaluation of each term in the document and not the document as a whole, we use the method described in Section 3.1.1, rather than using the *scoring queries as trees* method of Section 3.1.2. Hence we iterate through each suffix in the document and evaluate each in turn. The evaluation of each suffix is broken down into the evaluation of each term in the context of that suffix. And each term then gains a contribution to its total score from each of the suffixes it is a part of; the sum of all the scores the term receives is its total score, which is then taken as the indication of its significance. The details are shown in Algorithm 3.1 below.

3.1: Document-to-Class Term Significance Algorithm

```
Input   : Class tree, $T$; document $q$; a scoring method, score($a_j|A$), which scores the $j^{th}$ term, $a_j$
          in the string $A$.
Output: Array, $G$, of word significance scores.
Requirements: split($d$) to split a document $d$ into words.
1. Array of words, $A \leftarrow$ split($q$).
2. $d \leftarrow$ maxDepth($T$).
3. $n \leftarrow$ length($A$)
4. Declare score array, $G$, of length $n$.
5. Foreach suffix $A_i$ of $A$, for $i = [0, n-1]$
6.     Foreach word $a_j$ in $A_i$, for $j = [0, d]$
7.         $G[i+j] = G[i+j]$ + score($a_j|A_i$)        $(\phi[p(a_j|A_i)])$
8.     End Foreach
9. End Foreach
```

The output array, G, of the Document-to-Class Term Significance algorithm provides a significance score for each word in the document. The scores are partitioned into a number of score groups and words are highlighted according to which partition their score belongs and the highlighting policy we adopt. For example, if we highlight by two different colours (red for important, black for not important), we partition the scores at the median score and all those terms which scored above the median are highlighted in red and all those below are left at their original (usually black) font colour. Alternatively, we might highlight by font size, such that the higher the partition a term belongs to, the larger its font size. We are of course free to choose any arbitrary number of partitions, and we need not highlight all of them; so for example, we could divide the scores in four partitions, but only the highlighted in red the highest scoring partition (or in this case, quartile). The number of partitions and the highlighting policy itself is not a matter we address in the paper, but instead we present some examples using font size highlighting in Chapter 6.

Chapter 4

Application to Spam Filtering

This chapter describes the initial work we did to investigate the viability of using the suffix tree text model in classification tasks. We investigate the effect on classifier performance of altering parameters in the scoring method (Section 4.3; and in the experimental set up (Section 4.4.1. Performance is compared to the well known and extensively studied naive Bayes (NB) classifier – Section 4.2. Where possible, the comparison of results is done against results reported in the literature. However, we also include results from experiments on data sets which we specially collected during our research and for this reason we construct an in-house NB classifier in order to test how the performance of NB fairs against the suffix tree in these newly collect data sets.

The term *spam* refers to unsolicited emails. Such emails are well known by anyone with an email account, but we give some examples of types of spam in Section 4.1 in order to draw the reader's attention to specific characteristics which are useful to our argument; however, the examples are by no means comprehensive nor exhaustive.

Spam filtering represents a two-class problem with characteristics which may specifically benefit from the kind of analysis that may be performed using the the suffix tree text model. The key benefit lies in the suffix trees ability to make partial string matches; hence we are able to compare not only single words, as in most current approaches, but substrings of an arbitrary length. Comparisons of substrings (at the level of characters) has particular benefits in the domain of spam classification because of the methods spammers use to evade filters. For example, they may disguise the nature of their messages by interpolating them with meaningless characters, thereby fooling filters based on keyword features into considering the words, sprinkled with random characters, as completely new and unencountered. If we instead treat the words as character strings, we are still able to recognise the substrings, even if the words are broken.

This chapter is divided in the following way. Section 4.1 gives examples of some of the methods spammers use to evade detection which make it useful to consider character level features. Section 4.2 gives a brief explanation of the naive Bayes method of text classification as an example of a conventional approach. Section 4.4 describes our experiments, the test parameters and details of the data sets we used. Section 4.5 presents the results of the experiments and provides a comparison with results in the literature. Section 4.6 concludes this chapter.

4.1 Examples of Spam

Spam messages typically advertise a variety of products or services ranging from prescription drugs or cosmetic surgery to sun glasses or holidays. But regardless of what is being advertised, one can distinguish between the methods used by the spammer to evade detection. These methods have evolved with the filters which attempt to intercept them, so there is a generational aspect to them, with later generations becoming gradually more common and earlier ones fading out; as this happens, earlier generations of filters become less effective.

We present four examples of spam messages, the first of which illustrates undisguised spam while the other three illustrate one or more methods of evasion.

1. **Undisguised message.** The example contains no obfuscation. The content of the message is easily identified by filters, and words like “Viagra” allow it to be recognised as spam. Such messages are very likely to be caught by the simplest word-based Bayesian classifiers.

> Buy cheap medications online, no prescription needed.
>
> We have Viagra, Pherentermine, Levitra, Soma, Ambien, Tramadol and many more products.
>
> No embarrasing trips to the doctor, get it delivered directly to your door.
>
> Experienced reliable service.
>
> Most trusted name brands.
>
> Your solution is here: http://www.webrx-doctor.com/?rid=1000

2. **Intra-word characters.**

> Get the low.est pri.ce for gen.eric medica.tions!
>
> Xa.n.ax - only $100
>
> Vi.cod.in - only $99
>
> Ci.al.is - only $2 per do.se
>
> Le.vit.ra - only $73
>
> Li.pit.or - only $99
>
> Pr.opec.ia - only $79
>
> Vi.agr.a - only $4 per do.se
>
> Zo.co.r - only $99
>
> Your Sav.ings 40% compared Average Internet Pr.ice!
>
> No Consult.ation Fe.es! No Pr.ior Prescrip.tions Required! No Ap-poi.ntments! No Wait.ing Room! No Embarra.ssment! Private and Con-fid.ential! Disc.reet Packa.ging!
>
> che ck no w: `<http://priorlearndiplomas.com/r3/?d=getanon>`

The example above shows the use of intra-word characters, which may be non-alpha-numeric or whitespace. Here the word, “Viagra” has become “Vi.agr.a”, while the word “medications” has become “medica.tions”. To a simple word-based Bayesian classifier, these are completely new words, which might have occurred rarely, or not at all, in previous examples. Obviously, there are a large number of variations on this theme which would each time create an effectively new word which would not be recognised as spam content. However, if we approach this email at the character level, we can still recognise strings such as “medica” as indicative of spam, regardless of the character that follows, and furthermore, though we do not deal with this in the current paper, we might implement a look-ahead window which attempts to skip (for example) non-alphabetic characters when searching for spammy features.

Certainly, one way of countering such techniques of evasion is to map the obfuscated words to genuine words during a pre-processing stage, and doing this will help not only word-level filters, but also character-level filters because an entire word match, either as a single unit, or a string of characters, is better than a partial word match. The subject of spam de-obfuscation has been addressed in a recent paper by Lee and Ng 2005 [28].

However, some other methods may not be evaded so easily in the same way, with each requiring its own special treatment; we give two more examples below which illustrate the point.

3. **Word salad.**

Buy meds online and get it shipped to your door Find out more here `<http://www.gowebrx.com/?rid=1001>`

a publications website accepted definition. known are can Commons the be definition. Commons UK great public principal work Pre-Budget but an can Majesty’s many contains statements statements titles (eg includes have website. health, these Committee Select undertaken described may publications

The example shows the use of what is sometimes called a *word salad* - meaning a random selection of words. The first two lines of the message are its real content; the paragraph below is a paragraph of words taken randomly from what might have been a government budget report. The idea is that these words are likely to occur in ham, and would lead a traditional algorithm to classify this email as such. Again, approaching this at the character level can help. For example, say we consider strings of length 8, strings such as “are can” and “an can”, are unlikely to occur in ham, but the words “an”, “are” and “can” may occur quite frequently. Of course, in most ’bag-of-words’ implementations, words such as these are pruned from the feature set, but the argument still holds for other bigrams.

4. **Embedded message (also contains a word/letter salad).** (See example overleaf) The example below shows an *embedded* message. Inspection of it will reveal that it is actually offering prescription drugs. However, there are no easily recognised words, except those that form the word salad, this time taken from what appear to be dictionary entries under ’z’. The value of substring searching is highly apparent in this case as it allows us to recognise words such as “approved”, “Viagra” and “Tablets”, which would otherwise be lost among the characters pressed up against them.

These examples are only a sample of all the types of spam that exist, for an excellent and often updated list of examples and categories, see [19, 11]. Under the categories suggested in Wittel and Wu 2004 [60], example 2 and 4 would count as ’Tokenisation’ and/or ’Obfuscation’, while examples 2 and 3 would count as ’Statistical’.

We look next at a bag-of-words approach, naive Bayes, before considering the suffix tree approach.

zygotes zoogenous zoometric zygosphene zygotactic zygoid zucchettos zymolysis zoopathy zygophyllaceous zoophytologist zygomaticoauricular zoogeologist zymoid zoophytish zoospores zygomaticotemporal zoogonous zygotenes zoogony zymosis zuza zoomorphs zythum zoonitic zyzzyva zoophobes zygotactic zoogenous zombies zoogrpahy zoneless zoonic zoom zoosporic zoolatrous zoophilous zymotically zymosterol

FreeHYSHKRODMonthQGYIHOCSupply.IHJBUMDSTIPLIBJT

```
* GetJIIXOLDViagraPWXJXFDUUTabletsNXZXVRCBX
<http://healthygrow.biz/index.php?id=2>
```

zonally zooidal zoospermia zoning zoonosology zooplankton zoochemical zoogloeal zoological zoologist zooid zoosphere zoochemical

```
& Safezoonal andNGASXHBPnatural
& TestedQLOLNYQandEAVMGFCapproved
```

zonelike zoophytes zoroastrians zonular zoogloeic zoris zygophore zoograft zoophiles zonulas zygotic zymograms zygotene zootomical zymes zoodendrium zygomata zoometries zoographist zygophoric zoosporangium zygotes zumatic zygomaticus zorillas zoocurrent zooxanthella zyzzyvas zoophobia zygodactylism zygotenes zoopathological noZFYFEPBmas `<http://healthygrow.biz/remove.php>`

4.2 Naive Bayesian Classification

Naive Bayesian (NB) email filters currently attract a lot of research and commercial interest, and have proved highly successful at the task; Sahami et al. 1998 [49] and Androutsopoulos et al. 2000 [1] are both excellent studies of this approach to email filtering. We do not give detailed attention to NB; for a general discussion of NB see Lewis 1998 [30], for more context in text categorisation see Sebastiani 2002 [52], and for an extension of NB to the classification of structured data, see Flach and Lachiche 2005 [17]. However, an NB classifier is useful in our investigation of the suffix tree classifier, and in particular, our own implementation of NB is necessary to investigate experimental conditions which have not been explored in the literature. We therefore briefly present it here.

We begin with a set of training examples with each example document assigned to one of a fixed set of possible classes, $\mathbf{C} = \{\mathrm{c}_1, \mathrm{c}_2, \mathrm{c}_3, \ldots \mathrm{c}_J\}$. An NB classifier uses this training data to generate a probabilistic model of each class; and then, given a new document to classify, it uses the class models and Bayes' rule to estimate the likelihood with which each class generated the new document. The document is then assigned to the most likely class. The features, or parameters, of the model are usually individual words; and it is 'naive' because of the simplifying assumption that, given a class, each parameter is independent of the others.

McCallum and Nigam 1998 [36] distinguish between two types of probabilistic models which are commonly used in NB classifiers: the *multi-variate Bernoulli* event model and the *multinomial* event model. We adopt the latter, under which a document is seen as a series of word events and the probability of the event given a class is estimated from the frequency of that word in the training data of the class.

Hence, given a document $\mathbf{d} = \{d_1 d_2 d_3 \ldots d_L\}$, we use Bayes' theorem to estimate the probability of a class, c_j:

$$P(c_j \mid \mathbf{d}) = \frac{P(c_j)P(\mathbf{d} \mid c_j)}{P(\mathbf{d})} \tag{4.1}$$

Assuming that words are independent given the category, this leads to:

$$P(c_j \mid \mathbf{d}) = \frac{P(c_j)\prod_{i=1}^{L} P(d_i \mid c_j)}{P(\mathbf{d})} \tag{4.2}$$

We estimate P(c$_j$) as:

$$\hat{P}(C = c_j) = \frac{N_j}{N} \tag{4.3}$$

and P(d$_i$ | c$_j$) as:

$$\hat{P}(d_i \mid c_j) = \frac{1 + N_{ij}}{M + \sum_{k=1}^{M} N_{kj}} \tag{4.4}$$

where N is the total number of training documents, N_j is the total number of training documents belonging to class j, N_{ij} is the number of times word i occurs in class j (similarly for N_{kj}) and M is the total number of words considered.

To classify a document we calculate two scores, for spam and ham, and take the ratio, $hsr = \frac{hamScore}{spamScore}$, and classify the document as ham if it is above a threshold, th, and as spam if it is below (see Section 4.4.1).

4.3 Annotated Suffix Tree Classification

We experiment with two scoring parameters: match normalisation and the significance function; we deal with these below in the next section.

4.3.1 Match Normalisation

For match normalisation, we experiment here with three specifications of $v(m|T)$:

$$v(m|T) = \begin{cases} 1 & match\ \ unnormalised \\ \frac{f(m|T)}{\sum_{i \in (m^*|T)} f(i|T)} & match\ \ permutation\ \ normalised \\ \frac{f(m|T)}{\sum_{i \in (m'|T)} f(i|T)} & match\ \ length\ \ normalised \end{cases}$$

where m^* is the set of all the strings in T formed by the permutations of the letters in m; and m' is the set of all strings in T of length equal to the length of m.

Match permutation normalisation (MPN) is motivated by heuristic H2. The more diverse a class (meaning that it is represented by a relatively large set of substring features), the more combinations of characters we would expect to find, and so finding the particular match *m* is less significant than if the class were very narrow (meaning that it is fully represented by a relatively small set of substring features). Reflecting this, the MPN parameter will tend towards 1 if the class is less diverse and towards 0 if the class is more diverse.

Match length normalisation (MLN) is motivated by examples from standard linear classifiers (see Lewis et al. 1996 [31] for an overview), where length normalisation of feature weights is not uncommon. However, MLN actually runs counter to heuristic H1 because it will tend towards 0 as the match length increases. We would therefore expect MLN to reduce the performance of the classifier; thus MLN may serve as a test of the intuitions governing heuristic H1.

4.3.2 Significance Function

Recall that in the *overlap mass score*, a function of probability, $\phi[\hat{p}]$, is employed as a *significance function* because it is not always the most frequently occurring terms or strings which are most indicative of a class. For example, this is the reason that conventional pre-processing removes all stop words, and the most and least frequently occurring terms; however, by removing them completely we give them no significance at all, when we might instead include them, but reduce their significance in the classification decision. Functions on the probability can help to do this, especially in the absence of all pre-processing, but that still leaves the question of *how* to weight the probabilities, the answer to which will depend on the class.

In the spam domain, some strings will occur very infrequently (consider some of the strings resulting from intra-word characters in the examples of spam in Section 4.1 above) in either the spam or ham classes, and it is because they are so infrequent that they are indicative of spam. Therefore, under such an argument, rather than remove such terms or strings, we should actually *increase* their weighting.

Consideration such as these led us to experiment on spam filtering using the following definitions of the significance function: :

$\phi[\hat{p}]$:

$$\phi[\hat{p}] = \begin{cases} 1 & constant \\ \hat{p} & linear \\ \hat{p}^2 & square \\ \sqrt{\hat{p}} & root \\ \ln(\hat{p}) - \ln(1-\hat{p}) & logit \\ \frac{1}{1+\exp(-\hat{p})} & sigmoid \end{cases}$$

The first three functions after the constant are variations of the linear (linear, sub-linear and super-linear). The last two are variations on the S-curve; we give above the simplest forms of the functions, but in fact, they must be adjusted to fit in the range [0,1].

Although in this chapter we are not aiming to investigate a probabilistic scoring method, note that the logistic significance function may be considered an approximation of such an approach since we generally have a large alphabet and therefore a large number of children at each node, and so for most practical purposes $\ln(1-\hat{p}) \approx 0$.

4.4 Experimental Setup

All experiments were conducted under ten-fold cross validation. We accept the point made by Meyer and Whateley 2004 [38] that such a method does not reflect the way classifiers are used in practice, but the method is widely used and serves as a thorough initial test of new approaches.

We follow convention by considering as true positives (TP), spam messages which are correctly classified as spam; false positives (FP) are then ham messages which are incorrectly classified as spam; false negatives (FN) are spam incorrectly classified as ham; true negatives (TN) are ham messages correctly classified as ham. See Section 4.4.3 for more on the performance measurements we use.

4.4.1 Experimental Parameters

In addition to the various flavours of the scoring function, we experiment with the following experimental parameters.

Spam to Ham Ratios

From some initial tests we found that success was to some extent contingent on the proportion of spam to ham in our data set – a point which is identified, but not systematically investigated in other work [38] – and this therefore became part of our investigation. The differing results further prompted us to introduce forms of normalisation, even though we had initially expected the probabilities to take care of differences in the scale and mix of the data. Our experiments used three different ratios of spam to ham: 1:1, 4:6, 1:5. The first and second of these (1:1 and 4:6) were chosen to reflect some of the estimates made in the literature of the actual proportions of spam in current global email traffic. The last of these (1:5) was chosen as the minimum proportion of spam included in experiments detailed in the literature, for example in Androutsopoulos et al. 2000 [1].

Tree Depth

It is too computationally expensive to build trees as deep as emails are long. Furthermore, the marginal performance gain from increasing the depth of a tree, and therefore the length of the substrings we consider, may be negative. Certainly, our experiments show a diminishing marginal improvement (see Section 4.5.2), which would suggest a maximal performance level, which may not have been reached by any of our trials. We experimented with depths of length of 2, 4, 6, and 8.

Threshold

From initial trials, we observed that the choice of threshold value in the classification criterion can have a significant, and even critical, effect on performance, and so introduced it as an important experimental parameter. We used a range of threshold values between 0.7 and 1.3, with increments of 0.1, with a view to probing the behaviour of the scoring system.

Varying the threshold is equivalent to associating higher costs with either false positives or false negatives because checking that $(\alpha/\beta) > t$ is equivalent to checking that $\alpha > t\beta$.

4.4.2 Data

Text Corpora

Three corpora were used to create the training and testing sets:

1. The Ling-Spam corpus (LS)

   This is available from: `http://www.aueb.gr/users/ion/data/lingspam_public.tar.gz`. The corpus is that used in Androutsopoulos et al. 2000 [1]. The spam messages of the corpus were collected by the authors from emails they received. The ham messages are taken from postings on a public online linguist bulletin board for professionals; the list was moderated, so does not contain any spam. Such a source may at first seem biased, but the authors claim that this is not the case. There are a total of 481 spam messages and 2412 ham messages, with each message consisting of a subject and body.

   When comparing our results against those of Androutsopoulos et al. 2000 [1] in Section 4.5.1 we use the complete data set, but in further experiments, where our aim was to probe the properties of the suffix tree approach and investigate the effect of different proportions of spam to ham messages, we use a random subset of the messages so that the sizes and ratios of the experimental data sets derived from this source are the same as data sets made up of messages from other sources (see Table 4.1 below).

2. Spam Assassin public corpus (SA)

   This is available from: `http://spamassassin.org/publiccorpus`. The corpus was collected from direct donations and from public forums over two periods in 2002 and 2003, of which we use only the later. The set from 2003 comprise a total of 6053 messages, approximately 31% of which are spam. The ham messages are split into ’easy ham’ (SAe) and ’hard ham’ (SAh), the former being again split into two groups (SAe-G1 and SAe-G2); the spam is similarly split into two groups (SAs-G1 and SAs-G2), but there is no distinction between hard and easy. The compilers of the corpus describe hard ham as being closer in many respects to typical spam: use of HTML, unusual HTML markup, coloured text, “spammish-sounding” phrases etc..

   In our experiments we use ham from the hard group and the second easy group (SAe-G2); for spam we use only examples from the second group (SAs-G2). Of the hard ham there are only 251 emails, but for some of our experiments we required more examples, so whenever necessary we padded out the set with randomly selected examples from group G2 of the easy ham (SAe-G2); see Table 4.1. The SA corpus reproduces all header information in full, but for our purposes, we extracted the subjects and bodies of each; the versions we used are available at: `http://dcs.bbk.ac.uk/~rajesh/spamcorpora/spamassassin03.zip`

3. The BBKSpam04 corpus (BKS)
   This is available at: `http://dcs.bbk.ac.uk/~rajesh/spamcorpora/bbkspam04.zip`. This corpus consists of the subjects and bodies of 600 spam messages received by the authors during 2004. The Birkbeck School of Computer Science and Information Systems uses an installation of the SpamAssassin filter [2] with default settings, so all the spam messages in this corpus have initially evaded that filter. The corpus is further filtered so that no two emails share more than half their substrings with others in the corpus. Almost all the messages in this collection contain some kind of obfuscation, and so more accurately reflect the current level of evolution in spam.

One experimental *email data set* (EDS) consisted of a set of spam and a set of ham. Using messages from these three corpora, we created the EDSs shown in Table 4.1. The final two numbers in the code for each email data set indicate the mix of spam to ham; three mixes were used: 1:1, 4:6, and 1:5. The letters at the start of the code indicate the source corpus of the set's spam and ham, respectively; hence the grouping. For example, EDS SAe-46 is comprised of 400 spam mails taken from the group SAs-G2 and 600 ham mails from the group SAe-G2, and EDS BKS-SAeh-15 is comprised of 200 spam mails from the BKS data set and 1000 ham mails made up of 800 mails from the SAe-G2 group and 200 mails from the SAh group.

Pre-processing

For the suffix tree classifier, no pre-processing is done. It is likely that some pre-processing of the data may improve the performance of an ST classifier, but we do not address this issue in the current paper.

For the the naive Bayesian classifier, we use the following standard three pre-processing procedures:

1. Remove all punctuation.

2. Remove all stop-words.

EDS Code	Spam Source (number from source)	Ham Source (number from source)
LS-FULL	LS (481)	LS (2412)
LS-11	LS (400)	LS (400)
LS-46	LS (400)	LS (600)
LS-15	LS (200)	LS (1000)
SAe-11	SAs-G2 (400)	SAe-G2 (400)
SAe-46	SAs-G2 (400)	SAe-G2 (600)
SAe-15	SAs-G2 (200)	SAe-G2 (1000)
SAeh-11	SAs-G2 (400)	SAe-G2 (200) + SAh (200)
SAeh-46	SAs-G2 (400)	SAe-G2 (400) + SAh (200)
SAeh-15	SAs-G2 (200)	SAe-G2 (800) + SAh (200)
BKS-LS-11	BKS (400)	LS (400)
BKS-LS-46	BKS (400)	LS (600)
BKS-LS-15	BKS (200)	LS (1000)
BKS-SAe-11	BKS (400)	SAe-G2 (400)
BKS-SAe-46	BKS (400)	SAe-G2 (600)
BKS-SAe-15	BKS (200)	SAe-G2 (1000)
BKS-SAeh-11	BKS (400)	SAe-G2 (200) + SAh (200)
BKS-SAeh-46	BKS (400)	SAe-G2 (400) + SAh (200)
BKS-SAeh-15	BKS (200)	SAe-G2 (800) + SAh (200)

Table 4.1: Composition of Email Data Sets (EDSs) used in the experiments.

3. Stem all remaining words.

Words are taken as strings of characters separated from other strings by one or more whitespace characters (spaces, tabs, newlines). Punctuation is removed first in the hope that many of the intra-word characters which spammers use to confuse a Bayesian filter will be removed. Our stop-word list consisted of the 57 of the most frequent prepositions, pronouns, articles and conjunctives. Stemming was done using an implementation of Porter's 1980 algorithm [45], more recently reprinted in [46]. All words less than three characters long are ignored. For more general information on these and other approaches to pre-processing, the reader is directed to [35, 59].

4.4.3 Performance Measurement

There are generally two sets of measures used in the literature; here we introduce both in order that our results may be more easily compared with previous work.

Following Sahami et al. [49], Androutsopoulos et al. [1], and others, the first set of measurement parameters we use are *recall* and *precision* for both spam and ham. For spam (and similarly for ham) these measurements are defined as follows:

$$Spam\ \ Recall\ \ (SR) = \frac{SS}{SS+SH}\ ,\ \ Spam\ \ Precision\ \ (SP) = \frac{SS}{SS+HS}$$

; where XY means the number of items of class X assigned to class Y; with S standing for spam and H for ham.

Spam recall measures the proportion of all spam messages which were identified as spam and spam precision measures the proportion of all messages classified as spam which truly are spam; and similarly for ham.

However, it is now more popular to measure performance in terms of *true positive* (TP) and *false positive* (FP) rates:

$$TPR = \frac{SS}{SS+SH}\ , \qquad FPR = \frac{HS}{HH+HS}$$

The TPR is then the proportion of spam correctly classified as spam and the FPR is the proportion of ham incorrectly classified as spam. Using these measures, we plot in Section 4.5 what are generally referred to as receiver operator curves (ROC) [16] to observe the behaviour of the classifier at a range of thresholds.

To precisely see performance rates for particular thresholds, we also found it useful to plot, against threshold, false positive rates (FPR) and false negative rates (FNR):

$$FNR = 1 - TPR$$

Effectively, FPR measures errors in the classification of ham and FNR measures errors in the classification of spam.

	Pre-processing	No. of attrib.	th	SR(%)	SP(%)
NB	(a) bare	50	1.0	81.10	96.85
	(b) stop-list	50	1.0	82.35	97.13
	(c) lemmatizer	100	1.0	82.35	99.02
	(d) lemmatizer + stop-list	100	1.0	82.78	99.49
	(a) bare	200	0.1	76.94	99.46
	(b) stop-list	200	0.1	76.11	99.47
	(c) lemmatizer	100	0.1	77.57	99.45
	(d) lemmatizer + stop-list	100	0.1	78.41	99.47
	(a) bare	200	0.001	73.82	99.43
	(b) stop-list	200	0.001	73.40	99.43
	(c) lemmatizer	300	0.001	63.67	100.00
	(d) lemmatizer + stop-list	300	0.001	63.05	100.00
ST	bare	N/A	1.00	97.50	99.79
	bare	N/A	0.98	96.04	100.00

Table 4.2: Results of (Androutsopoulos et al., 2000) on the Ling-Spam corpus. In the pre-processing column: 'bare' indicates no pre-processing. The column labelled 'No. of attrib.' indicates the number of word features which the authors retained as indicators of class. (Androutsopoulos et al. 2000) quote a 'cost value', which we have converted into equivalent values comparable to our threshold; these converted values are presented, in the 'th' column, in a rounded format. Results are shown at the bottom of the table from ST classification using a linear significance function and no normalisation; for the ST classifier, we performed no pre-processing and no feature selection.

4.5 Results

We begin in Section 4.5.1 by comparing the results of the suffix tree (ST) approach to the reported results for a naive Bayesian (NB) classifier on the the Ling Spam corpus. We then extend the investigation of the suffix tree to other data sets to examine its behaviour under different conditions and configurations. To maintain a comparative element on the further data sets we implemented an NB classifier which proved to be competitive with the classifier performance as reported in Androutsopoulos et al. [1] and others. In this way we look at each experimental parameter in turn and its effect on the performance of the classifier under various configurations.

4.5.1 Assessment

Table 4.2 shows the results reported in Androutsopoulos et al. [1], from the application of their NB classifier on the LS-FULL data set, and the results of the ST classifier, using a linear significance function

	Pre-processing	No. of attrib.	th	SR(%)	SP(%)
NB*	lemmatizer + stop-list	unlimited	1.0	99.16	97.14
	lemmatizer + stop-list	unlimited	0.94	89.58	100.00

Table 4.3: Results of in-house nave Bayes on the LS-FULL data set, with stop-words removed and all remaining words lemmatized. The number of attributes was unlimited, but, for the LS-FULL data set, in practice the spam vocabulary was approximately 12,000, and the ham vocabulary approximately 56,000, with 7,000 words appearing in both classes.

with no normalisation, on the same data set.

As can be seen, the performance levels for precision are comparable, but the suffix tree simultaneously achieves much better results for recall.

Androutsopoulos et al. [1] test a number of thresholds (*th*) and found that their NB filter achieves a 100% spam precision (SP) at a threshold of 0.001. We similarly tried a number of thresholds for the ST classifier, as previously explained (see Section 4.4.1), and found that 100% SP was achieved at a threshold of 0.98. Achieving high SP comes at the inevitable cost of a lower spam recall (SR), but we found that our ST can achieve the 100% in SP with less cost in terms of SR, as can be seen in the table.

As stated in the table (and previously: see Section 4.4.2), we did no pre-processing and no feature selection for the suffix tree. However, both of these may well improve performance, and we intend to investigate this in future work.

As we mentioned earlier (and in Section 4.2), we use our own NB classifier in our further investigation of the performance of our ST classifier. We therefore begin by presenting in Table 4.3 the results of this classifier (NB*) on the LS-FULL data set. As the table shows, we found our results were, at least in some cases, better than those reported in Androutsopoulos et al. [1]. This is an interesting result which we do not have space to investigate fully in this paper, but there are a number of differences in our naive Bayes method which may account for this.

Firstly, Androutsopoulos et al. [1] use a maximum of 300 attributes, which may not have been enough for this domain or data set, whereas we go to the other extreme of not limiting our number of attributes, which would normally be expected to ultimately reduce performance, but only against an optimal number, which is not necessarily the number used by Androutsopoulos et al. [1]. Indeed, some researchers [32, 64] have found NB does not always benefit from feature limitation, while others have

Classifier	Spam (%)	Ham (%)
NB$'$	96.47	99.34
NB*	94.96	98.82
ST	98.75	99.75

Table 4.4: Precision-recall breakeven points on the LS-FULL data set.

found the optimal number of features to be in the thousands or tens of thousands [51, 36]. Secondly, there may be significant differences in our pre-processing, such as a more effective stop-word list and removal of punctuation; and thirdly, we estimate the probability of word features using Laplace smoothing (see formula 4.4), which is more robust than the estimated probability quoted by Androutsopoulos et al. [1].

There may indeed be further reasons, but it is not our intension here to analyse the NB approach to text classification, but only to use it as a comparative aid in our investigation of the performance of the ST approach under various conditions. Indeed, other researchers have extensively investigated NB and for us to conduct the same depth of investigation would require a dedicated paper.

Furthermore, both our NB* and ST classifiers appear to be competitive with quoted results from other approaches using the same data set. For example in Schneider 2003 [51], the author experiments on the Ling-Spam data set with different models of NB and different methods of feature selection, and achieves results approximately similar to ours. Schneider [51] quotes "*breakeven*" points, defined as the "highest recall for which recall equaled precision", for both spam and ham; Table 4.4 shows the results achieved by the author's best performing naive Bayes configuration (which we label as 'NB$'$') alongside our naive Bayes (NB*) and the suffix tree (ST) using a linear significance function and no normalisation. As can be seen, NB* achieves slightly worse results than the NB$'$, while ST achieves slightly better results; but all are clearly competitive. And as a final example, Surkov 2004 [55] applies developments and extensions of support vector machine algorithms [57] to the Ling-Spam data set, albeit in a different experimental context, and achieves a minimum sum of errors of 6.42%; which is slightly worse than the results achieved by our NB* and ST classifiers.

Thus, let us proceed on the assumption that both our (NB* and ST) classifiers are at least competitive enough for the task at hand: to investigate how their performance *varies* under experimental conditions for which results are not available in the literature.

Depth	FPR(%)	FNR(%)
2	58.75	11.75
4	0.25	4.00
6	0.50	2.50
8	0.75	1.50

Table 4.5: Classification errors by depth using a constant significance function, with no normalisation, and a threhsold of 1 on the LS-11 email data set.

4.5.2 Analysis

In the following tables, we group email data sets (EDSs), as in Table 4.1, Section 4.4.2, by their source corpora, so that each of the EDSs in one group differ from each other only in the proportion of spam to ham they contain.

Effect of Depth Variation

For illustrative purposes, Table 4.5 shows the results using the *constant* significance function, with no normalisation using the LS-11 data set. Depths of 2, 4, 6, and 8 are shown.

The table demonstrates a characteristic which is common to all considered combinations of significance and normalisation functions: performance improves as the depth increases. Therefore, in further examples, we consider only our maximum depth of 8. Notice also the decreasing marginal improvement as depth increases, which suggests that there may exist a maximal performance level, which was not necessarily achieved by our trials.

Effect of Significance Function

We found that all the significance functions we tested worked very well, and all of them performed better than our naive Bayes. Figure 4.1 shows the ROC curves produced by each significance function (with no normalisation) for what proved to be one of the most difficult data sets (SAeh-11: see Section 4.4.2).

We found little difference between the performance of each of the functions across all the data sets we experimented with, as can be seen from the summary results in Table 4.6, which shows the minimum sum of errors (FPR+FNR) achieved at a threshold of 1.0 by each significance function on each data set. The constant function looks marginally the worst performer and the logit and root functions marginally the best, but this difference is partly due to differences in optimal threshold (see Section 4.4.1) for each

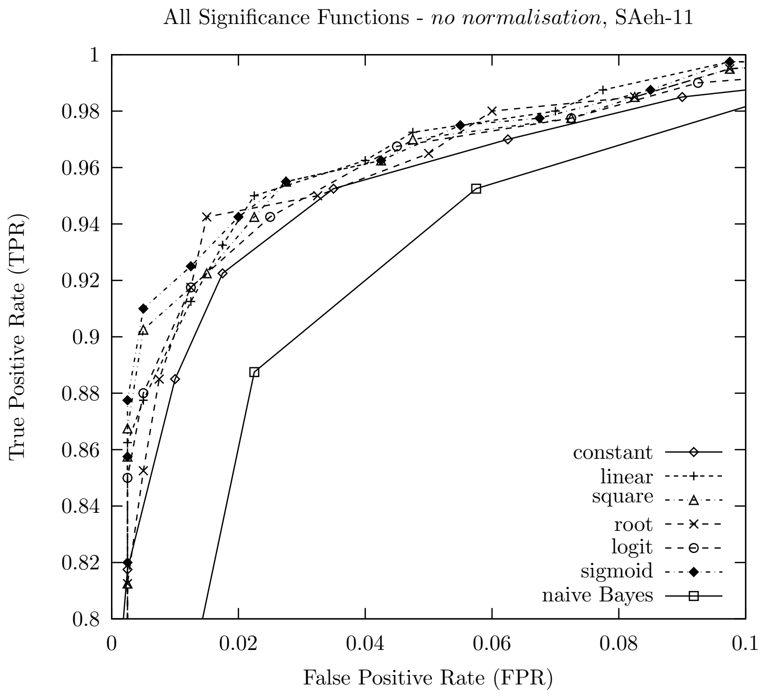


Figure 4.1: ROC curves for all significance functions on the SAeh-11 data set.

function: those that perform less well at a threshold of 1.0 may perform better at other thresholds.

Table 4.7 presents the minimum sum of errors achieved by each function at its individual optimal threshold. In this table there is even less difference between the functions, but still the root is marginally better than the others in that it achieves the lowest average sum of errors over all data sets. And so, for the sake of brevity we favour this function in much of our following analysis.

Effect of Threshold Variation

We generally found that there was an optimal threshold (or range of thresholds) which maximised the success of the classifier. As can be seen from the four example graphs shown in Figure 4.2, the optimal threshold varies depending on the significance function and the mix of ham and spam in the training and testing sets, but it tends to always be close to 1.

Obviously, it may not be possible to know the optimal threshold in advance, but we expect, though have not shown, that the optimal threshold can be established during a secondary stage of training where only examples with scores close to the threshold are used - similar to what Meyer and Whateley 2004

	Sum of Errors (%) at $th = 1$ for specifications of $\phi[\hat{p}]$					
EDS Code	constant	linear	square	root	logit	sigmoid
LS-11	**1.5**	2.25	2.5	1.75	1.75	1.75
LS-46	1.33	1.42	1.92	**1.08**	1.58	1.42
LS-15	**1.33**	**1.33**	1.55	**1.33**	1.55	1.89
SAe-11	**0.25**	0.5	0.5	0.5	**0.25**	0.75
SAe-46	0.5	0.75	0.5	0.5	**0.25**	0.75
SAe-15	**1.00**	1.50	1.8	1.5	1.1	2.00
SAeh-11	7.00	7.00	7.50	6.75	**5.49**	6.50
SAeh-46	**4.33**	4.58	4.92	5.00	4.42	4.92
SAeh-15	9.3	**7.5**	8.00	7.7	7.6	8.6
BKS-LS-11	**0**	**0**	**0**	**0**	**0**	**0**
BKS-LS-46	**0**	**0**	**0**	**0**	**0**	**0**
BKS-LS-15	**0**	1.5	1.5	1.00	**0**	1.5
BKS-SAe-11	4.75	1.75	**1.5**	**1.5**	**1.5**	1.75
BKS-SAe-46	4.5	1.75	2.00	1.75	**1.5**	2.75
BKS-SAe-15	9.5	6.00	6.00	**5.50**	**5.50**	8.5
BKS-SAeh-11	9.25	5.75	7.25	**5.00**	5.75	7.25
BKS-SAeh-46	10.25	5.25	7.00	**4.25**	5.00	7.25
BKS-SAeh-15	15.5	**9.5**	**9.5**	**9.5**	**9.5**	14.5

Table 4.6: Sum of errors (FPR+FNR) values at a conventional threshold of 1 for all significance functions under match permutation normalisation (see Section 6.2.4). The best scores for each email data set are highlighted in bold.

[38] call “non-edge training”.

In any case, the main reason for using a threshold is to allow a potential user to decide the level of false positive risk they are willing to take: reducing the risk carries with it an inevitable rise in false negatives. Thus we may consider the lowering of the threshold as attributing a greater cost to miss-classified ham (false positives) than to miss-classified spam; a threshold of 1.0 attributes equal importance to the the two.

Figure 4.2 Graphs (a-c) show three different combinations of significance function and data set, the shapes of which are representative of all combinations. The performance of a particular scoring configuration is reflected not only by the minimums achieved at optimal thresholds but also by the

	Sum of Errors (%) at optimal th for specifications of $\phi[\hat{p}]$					
EDS Code	constant	linear	square	root	logit	sigmoid
LS-11	1.25	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
LS-46	1.08	1.08	1.00	1.08	1.08	**0.83**
LS-15	**1.33**	**1.33**	**1.33**	**1.33**	**1.33**	**1.33**
SAe-11	0.25	**0**	**0**	**0**	**0**	0.25
SAe-46	0.42	0.33	0.33	**0.25**	**0.25**	0.5
SAe-15	**1.00**	1.3	1.4	1.1	1.1	1.2
SAeh-11	7.00	6.50	**6.00**	6.25	6.25	6.50
SAeh-46	**4.00**	4.58	4.92	4.42	4.33	4.92
SAeh-15	6.50	6.60	6.70	6.50	6.60	**6.30**
BKS-LS-11	**0**	**0**	**0**	**0**	**0**	**0**
BKS-LS-46	**0**	**0**	**0**	**0**	**0**	**0**
BKS-LS-15	**0**	**0**	**0**	**0**	**0**	**0**
BKS-SAe-11	**0**	**0**	**0**	**0**	**0**	**0**
BKS-SAe-46	**0**	**0**	**0**	**0**	**0**	**0**
BKS-SAe-15	0.2	**0**	**0**	**0**	0.2	0.50
BKS-SAeh-11	2.75	**1.75**	2.00	**1.75**	2.00	2.00
BKS-SAeh-46	1.33	1.17	1.50	1.17	**1.00**	1.33
BKS-SAeh-15	**1.1**	1.2	2.00	1.30	**1.1**	2.1

Table 4.7: Sum of error (FPR+FNR) values at individual optimal thresholds for all significance functions under match permutation normalisation (see Section 6.2.4). The best scores for each data set are highlighted in bold.

steepness (or shallowness) of the curves: the steeper they are, the more rapidly errors rise at sub-optimal levels, making it harder to achieve zero false positives without a considerable rise in false negatives. Graph (d) shows that our NB classifier is the most unstable in this respect.

Effect of Normalisation

We found that there was a consistent advantage to using *match permutation* normalisation, which was able to improve overall performance as well as making the AST classifier more stable under varying thresholds. Figure 4.3 shows the ROC curves produced by the constant significance function under match permutation normalisation (MPN); match length normalisation (MLN) reduced performance so

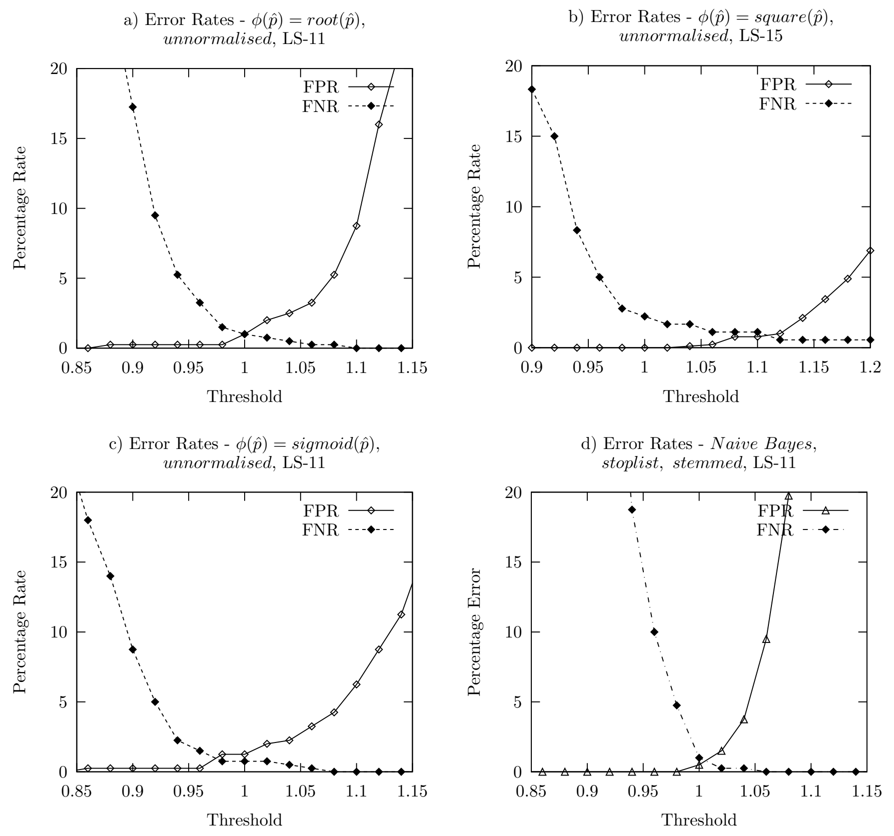


Figure 4.2: Effect of threshold variation. Graphs (a-c) show suffix tree false positive (FP) and false negative (FN) rates for three specifications of $\phi(\hat{p})$ under no normalisation; graph (d) shows naive Bayes FP and FN rates.

much that the resulting curve does not even appear in the range of the graph. The stabilising effect of match permutation normalisation is reflected in ROC curves by an increase in the number of points along the curve, but may be better seen in Figure 4.4 as a shallowing of the FPR and FNR curves. The negative effect of MLN concurs with our heuristics from Section 3.1, as we conjectured it would in Section 4.3.1. These effects of MPN and MLN were observed, to a greater or lesser degree, for all significance functions and data sets.

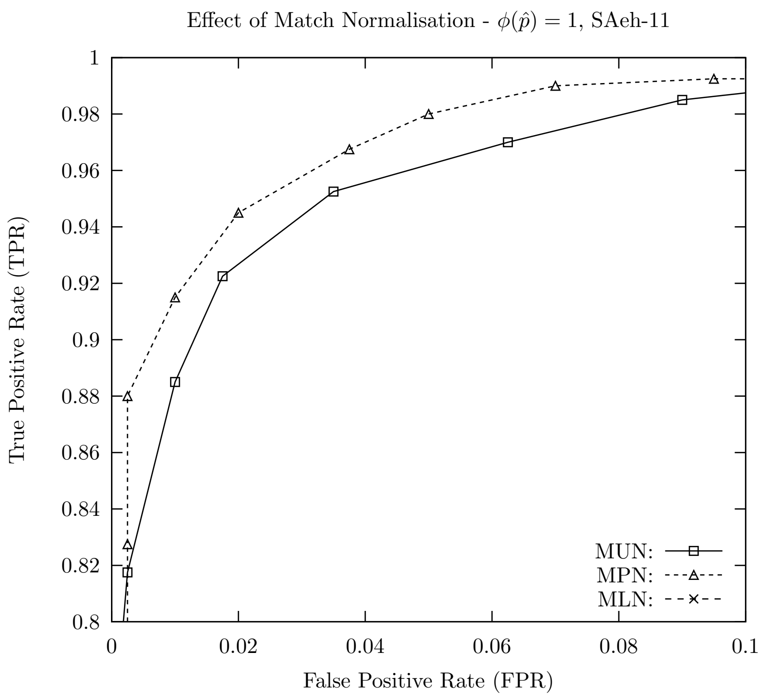


Figure 4.3: ROC curves for the constant significance function under no match normalisation (MUN), match permutation normalisation (MPN) and match length normalisation (MLN), on the SAeh-11 data set. MLN has such a detrimental effect on performance that its ROC curve is off the scale of the graph.

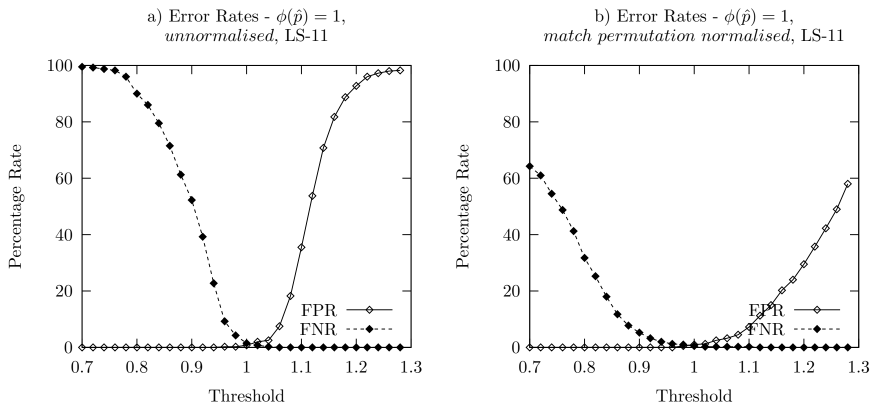


Figure 4.4: Effect of match permutation normalisation. False positive (FP) and false negative (FN) rates using a constant significance function on the LS-11 EDS. Graph (a) shows the false positive (FP) and false negative (FN) rates under no normalisation and graph (b) shows FP and FN rates under match permutation normalisation.

Effect of Spam to Ham Ratios

We initially found that the mix of spam to ham in the data sets could have some effect on performance, with the degree of difference in performance depending on the data set and the significance function used; however, with further investigation we found that much of the variation was due to differences in the optimal threshold. This can be seen by first examining the differences in performance for different spam:ham ratios shown in Table 4.6, in which a 1:5 ratio appears to result in lower performance than the more balanced ratios of 4:6 and 1:1; then examining the results presented in Table 4.7, where differences are far less apparent. These observations are reinforced by the graphs shown in Figure 4.5. In graph (a) which shows the ROC curves produced by the constant significance function with no normalisation on the SAeh data sets, we can see that the curves produced by different ratios appear to achieve slightly different maximal performance levels but roughly follow the the same pattern. Graphs (b-d) further show that the maximal levels of performance are achieved at different threshold for each ratio. The difference between Graph (b) and Graph (c) is not great as the difference between the ratios (1:1 and 4:6) is not great; however, the curves shift slightly to the right. The shift is more apparent in Graph (d) because there is a greater change in the ratio (from 1:1 to 1:5).

Overall Performance Across Email Data Sets

Table 4.8 summarises the results for both the ST and NB classifiers at a threshold of 1.0 and Table 4.9 summarises results at the individual optimal thresholds which minimise the sum of the errors (FPR+FNR).

We found that the performance of the NB is in some cases dramatically improved at its optimal threshold, for example in the case of the BKS-LS data sets. But at both a threshold of 1.0 and at optimal thresholds, the NB classifier behaves very much as expected, supporting our initial assumptions as to the difficulty of the data sets. This can be clearly seen in Table 4.9: on the SAeh data sets which contain ham with ’spammy’ features, the NB classifier’s false positive rate increases, meaning that a greater proportion of ham has been incorrectly classified as spam; and on the BKS-SAeh data sets which additionally contain spam which is disguised to appear as ham, the NB classifier’s false negative rate increases, meaning that a greater proportion of spam has been misclassified as ham.

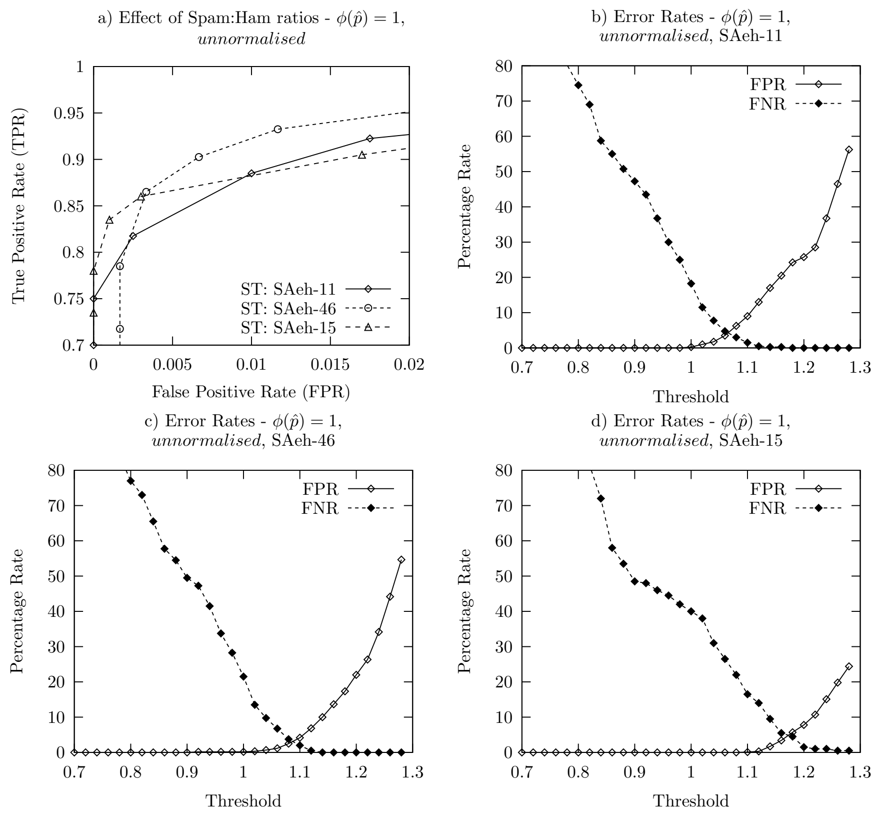


Figure 4.5: Effect of varying ratios of spam:ham on the SAeh data using a constant significance function with no normalisation. Graph (a) shows the ROC curves produced for each ratio; while graphs (b-d) show the FP and FN rates separately for ratios of 1:1, 4:6 and 1:5 respectively.

The performance of the ST classifier also improves at its optimal thresholds, though not so dramatically, which is to be expected considering our understanding of how it responds to changes in the threshold (see Section 4.5.2). The ST also shows improved performance on data sets involving BKS data. This may be because the character level analysis of the suffix tree approach is able to treat the attempted obfuscations as further positive distinguishing features, which do not exist in the more standard examples of spam which constitute the LS data sets. In all cases except on the SAeh data, the ST is able to keep the sum of errors close to or below 1.0, and in some cases, it is able to achieve a zero sum of errors. Furthermore, the suffix tree's optimal performance is often achieved at a range of thresholds, supporting our earlier observation of greater stability in it's classification success.

	Naive Bayes		Suffix Tree	
EDS Code	FPR (%)	FNR (%)	FPR (%)	FNR (%)
LS-11	1.25	0.50	1.00	1.75
LS-46	0.67	1.25	0.83	0.25
LS-15	1.00	1.00	0.22	0.13
SAe-11	0	2.75	0	0.50
SAe-46	0.17	2.00	0	0.50
SAe-15	0.30	3.50	0	1.50
SAeh-11	10.50	1.50	3.50	3.25
SAeh-46	5.67	2.00	2.00	3.00
SAeh-15	4.10	7.00	0.70	7.00
BKS-LS-11	0	12.25	0	0
BKS-LS-46	0.17	13.75	0	0
BKS-LS-15	0.20	30.00	0	1.00
BKS-SAe-11	0	9.00	0	1.50
BKS-SAe-46	0	8.25	0	1.75
BKS-SAe-15	1.00	15.00	0	5.5
BKS-SAeh-11	16.50	0.50	0	5.00
BKS-SAeh-46	8.17	0.50	0	4.25
BKS-SAeh-15	8.10	5.50	0	9.50

Table 4.8: Classification errors at threshold of 1, for Naive Bayes (NB) and a Suffix Tree (ST) using a root significance function and match permutation normalisation.

Computational Performance

For illustrative purposes, in this section we provide some indication of the time and space requirements of the annotated suffix tree (AST) classifier using a tree of depth, $d = 8$. However, it should be stressed that in our implementation of the AST classifiers (ASTCs) we made no attempts to optimise our algorithms as performance was not one of our concerns in this paper. The figures quoted here may therefore be taken as indicators of worst-case performance levels.

Table 4.10 summarises the time and space requirements of the ASTC on four of our email data sets. The AST approach clearly and unsurprisingly has high resource demands, far above the demands of a naive Bayes classifier which on the same machine typically uses no more than 40MB of memory and

	Naive Bayes			Suffix Tree		
EDS Code	OtpTh	FPR (%)	FNR (%)	OptTh	FPR (%)	FNR (%)
LS-11	1.0	1.25	0.50	0.96	0	1.00
LS-46	1.02	1.00	0.67	0.96	0.33	0.75
LS-15	1.00	1.00	1.00	0.98 - 1.00	0.22	1.11
SAe-11	1.06	0.25	0	1.10	0	0
SAe-46	1.04	0.33	0.25	1.02	0	0.25
SAe-15	1.02	2.30	1.50	1.02	0.1	1.00
SAeh-11	0.98	10.50	1.50	0.98	2.75	3.50
SAeh-46	1.00	5.67	2.00	0.98	1.16	3.25
SAeh-15	1.02	7.60	1.50	1.10	3.50	3.00
BKS-LS-11	1.04	0.75	2.25	0.78 - 1.22	0	0
BKS-LS-46	1.06	2.50	1.25	0.78 - 1.16	0	0
BKS-LS-15	1.10	5.50	1.50	1.02 - 1.22	0	0
BKS-SAe-11	1.04 - 1.06	0	0.25	1.04 - 1.28	0	0
BKS-SAe-46	1.06	0.50	0.25	1.18 - 1.28	0	0
BKS-SAe-15	1.04	6.90	0	1.20	0	0
BKS-SAeh-11	0.98	8.00	2.00	1.06	0	1.75
BKS-SAeh-46	0.98	4.00	3.75	1.14 - 1.16	0.67	0.5
BKS-SAeh-15	1.00	8.10	5.50	1.24 - 1.26	0.80	0.50

Table 4.9: Classification Errors at optimal thresholds (where the sum of the errors is minimised) for Naive Bayes (NB) and a Suffix Tree (ST) using a root significance function and match permutation normalisation.

EDS Code (size)	Training	AvSpam	AvHam	AvPeakMem
LS-FULL (7.40MB)	63s	843ms	659ms	765MB
LS-11 (1.48MB)	36s	221ms	206ms	259MB
SAeh-11 (5.16MB)	155s	504ms	2528ms	544MB
BKS-LS-11 (1.12MB)	41s	161ms	222ms	345MB

Table 4.10: Computational performance of suffix tree classification on four bare (no pre-processing) data sets. Experiments were run on a pentium IV 3GHz Windows XP laptop with 1GB of RAM. Averages are taken over all ten folds of cross-validation.

takes approximately 10 milliseconds (ms) to make a classification decision.

The difference in performance across the data sets is, however, exactly as we would expect considering our assumptions regarding them. The first point to note is that the mapping from data set size to tree size is non-linear. For example, the LS-FULL EDS is 5 times larger than the LS-11 EDS but results in a tree only 2.95 times larger. This illustrates the logarithmic growth of the tree as more information is added: the tree only grows to reflect the diversity (or complexity) of the training data it encounters and not the actual size of the data. Hence, though the BKS-LS-11 EDS is in fact approximately 25% smaller than the LS-11 data set, it results in a tree that is over 30% larger. We would therefore expect to eventually reach a stable maximal size once most of the complexity of the profiled class is encoded.

The current space and time requirements are viable, though demanding, in the context of modern computing power, but a practical implementation would obvious benefit from optimisation of the algorithms [1].

Time could certainly be reduced very simply by implementing, for example, a binary search over the children of each node; the search is currently done linearly over an alphabet of approximately 170 characters (upper- and lower- case characters are distinguished, and all numerals and special characters are considered; the exact size of the alphabet depends on the specific content of the training set). And there are several other similarly simple optimisations which could be implemented.

However, even with a fully optimised algorithm, the usual trade-off between resources and performance will apply. With regard to this, an important observation is that resource demands increase exponentially with depth, whereas performance increases logarithmically. Hence an important factor in any practical implementation will be the choice of the depth of the suffix tree profiles of classes.

4.6 Conclusion

Clearly, the non-parametric ASTC performs universally well across all the email data sets we experimented with, but there is still room for improvement: whereas in some cases, the approach is able to achieve perfect classification accuracy, this is not consistently maintained. Performance may be improved by introducing some pre-processing of the data or post-processing of the AST profile and it would be worthwhile investigating the possibility of improving performance on the current data sets,

[1]The literature on suffix trees deals extensively with improving (reducing) the resource demands of suffix trees [56, 18, 26].

but there is also the question of whether this method can be effectively applied in a broader text classification context. We investigate this possibility in the next chapters.

What we have so far argued and experimentally demonstrated is that at least in the domain of email filtering, the character-level approach to text modelling can be highly effective, particularly because it works to counter the character-level obfuscations frequently utilised in spam messages. Hence, we have confined our investigations exclusively to character-based models, using results from the literature to compare against more traditional word-based approaches. Next, we would now like to see if a character-level approach can still be effective when there are no such obfuscations. In fact, we would like to know how well a character-level approach performs in domains where word-based approaches have a natural advantage; and we do this in the next chapter when we experiment with the Reuters data set, a well know corpus that is normally thought to be particularly suited to word-based approaches. Hence, in what follows we experiment with word-based suffix trees to investigate (by contrast) just how useful this character-level approach is.

In the current chapter, we have further experimented with a number of simple, non-probabilistic significance functions and found that the root function performs marginally better than others, though the difference is small. This is explicable in the domain of email filtering because the probabilities (taken as weights) associated with a particular character-sequence tends to be quite small and a root significance function makes the classifier more sensitive to lower probabilities. In the next chapters we explore and compare this simple non-probabilistic significance function with the probabilistic significance function we developed in Section 3.1.2.

However, we have seen that the choice of significance function is the least important factor in the success of the AST approach because all of them performed acceptably well. Different functions will perform better on different data sets, but the root function appeared to perform marginally more consistently well on all the email data sets we experimented with.

Match permutation normalisation was found to be the most effective method of normalisation and was able to improve the performance of all significance functions. In particular it was able to improve the success of the filter at all threshold values. However, other methods of normalisation were not always so effective, with some of them making things drastically worse.

The threshold was found to be a very important factor in the success of the filter. So much so, that the differences in the performances of particular configurations of the filter were often attributable more to differences in their corresponding optimal thresholds than to the configurations themselves. However, as a cautionary note, variations in the optimal threshold may be due to peculiarities of the data sets involved, and this could be investigated further.

We also found that the false positive rate (FPR) and false negative rate (FNR) curves created by varying the threshold, were in all cases relatively shallower for our AST classifier than those for our NB classifier, indicating that the former always performs relatively better at non-optimal thresholds, thereby making it easier to minimise one error without a significant cost in terms of the other error.

Finally, any advantages in terms of accuracy in using the ASTC to filter emails, must be balanced against higher computational demands. In this paper, we have given little attention to minimising this factor, but even though available computational power tends to increase dramatically, cost will nevertheless be important when considering the development of the method into a viable email filtering application, and this is clearly a viable line of further investigation. However, the computational demands of the approach are not intractable, and an ASTC may be valuable in situations where accuracy is the primary concern.

Chapter 5

Application to Document Classification

In the previous chapter (Chapter 4) we investigated a number of aspects of the general classification methods described in Chapters 2-3 and established that the character-level suffix tree classifier can offer particular benefits over the traditional bag-of-word-based approaches, such as naive Bayes, in the domain of email filtering.

We now continue to develop and investigate aspects and configurations of the classifier framework and further extend the testing into a more general text classification domain involving multiple, possibly overlapping classes. Decision mechanisms to handle this multi-class case are introduced and the probabilistic significance function 3.1.1 is tested against the root function.

We further extend the scope of the classifier framework to include word-based class analysis and document classification. We show that the annotated suffix tree classification model we have described can be used effectively, not only as a character-based classifier, but also as a word-based classifier, in which each node represents a word and each path represents a sequence of words. We show that such a model can in many contexts outperform many of the other word-based approaches reported in the literature.

We do not, however, leave the character-based model behind. We show that the annotated suffix tree character-based model can in certain cases outperform many existing substring approaches described in the literature as well as compete with and outperform some word-based approaches, even in a more general text classification context.

Overall then, our aims in this chapter will be to investigate the following configurations and developments:

- Development of multi-class decision mechanisms.
- Comparison of probabilistic suffix tree classifiers, using the probabilistic significance function, (Section 3.1.2) against a simpler root significance function (Section 4.3.2).
- The difference in performance of a word-level and a character-level suffix tree classifier, each at varying depths.

To investigate each of these issues, we utilise the Reuters 21578 corpus of news articles – arguably the most popular benchmark text classification data set in wide usage. We take a closer look at this data set in the next section.

5.1 The Reuters Collection

The Reuters collection is a set of news stories assembled and indexed by personnel at Reuters Ltd in 1987, which were originally made available for research purposes in 1990. We use a version of this, the Reuters-21578 collection, which was further formatted and error corrected in 1991 and 1992; for full details see Lewis' distribution site [29]. We use the so-called "Modified Apte (ModApte) Split" of the data set which temporally separates the stories into 9603 training examples which were written and indexed before the 3299 testing examples. There are a total of 118 classes in the 'TOPIC' set and each story may belong to one or many of these classes.

The number of example documents in each class varies considerably, with some of the the smallest classes containing just one training example and no testing examples, while the the largest class, *earn*, contains 2877 training and 1087 testing examples. Around 10% of the documents belong to multiple classes and in many cases, the class(es) of a document may be predicted by the occurrence of a small, well-defined set of keywords [8, 23], which makes this data set highly suited to word-based classifiers.

In addition to class variation, individual stories can also vary greatly in their characteristics. Some are very long, with over 1000 words in the body of the story, while others may simply be headline

```
<REUTERS NEWID=15860>
  <TOPICS>
    <D>grain</D>
    <D>corn</D>
    <D>wheat</D>
  </TOPICS>
  <Title>USDA DETAILS FREE GRAIN STOCKS UNDER LOAN</Title>
  <Body>The U.S. Agriculture Department gave projected
  carryover free stocks of feedgrains, corn and wheat under
  loans, with comparisons, as follows, in mln bushels, except
  feedgrains, which is in mln tonnes --
                1986/87            1985/86
           04/09/87 03/09/87  04/09/87 03/09/87
 Under Regular Nine Month Loan --
      WHEAT    225     300       678     678
 FEEDGRAINS   52.1    68.1      75.7    75.7
       CORN  1,800   2,400     2,589   2,589
 Special Producer Storage Loan Program --
      WHEAT    165     150       163     163
 FEEDGRAINS    7.0     6.7       5.3     5.3
       CORN    200     200       147     147
   </Body>
```

Figure 5.1: Example Reuters News Story (top) belonging to the ‘grain’, ‘corn’ and ‘wheat’ classes.

stories consisting of only a title and no body. The formatting of stories can vary greatly too. For example, Figure 5.1 shows a story belonging to the ‘grain’, ‘corn’ and ‘wheat’ classes which reports grain stocks in tabular form. Similar tabular formats also occur frequently in other classes such as ‘earn’, in which company earnings are often reported. This tabular form is in contrast to stories with the more standard format demonstrated by the format of the story shown in Figure 5.2, which belongs to the ‘acq’ (acquisition) class. Such formatting can provide important clues to classification; and the character-based classifier, without pre-processing (e.g. the removal of punctuation), can make good use of these clues.

The Reuters data set is a particularly good one to use for our current investigation exactly because it is generally considered to be a data set that is particularly amenable to word-based analysis [8], in stark contrast to the spam data sets we used in the previous chapter, and so provides the character-based AST classifier with a challenging task.

```
<REUTERS NEWID=15161>
  <TOPICS>
    <D>acq</D>
  </TOPICS>
  <Title>U.K. GEC DECLINES COMMENT ON U.S. PURCHASE RUMOUR</Title>
  <Body>General Electric Co Plc <GECL.L> (GEC) declined comment on
  rumours on the London stock market that it is planning another
  purchase in the U.S. Medical equipment field, in addition to its
  existing U.S. Subsidiary <Picker International Inc>.
    A GEC spokesman said that it is company policy not to comment
  on acquisition rumours.
    Stock Exchange traders said the rumour helped GEC's share
  price to rise 5p, to a final 206p from yesterday's closing price
  of 201p.
  </Body>
```

Figure 5.2: Example Reuters News Story (top) belonging to the ‘acq’ class.

The data set is a well studied one with many researchers publishing results using various classification techniques, and so we do not, as we did in the previous chapter, implement any of our own versions of alternative techniques in order to compare the performance of the AST classifier, but simply compare its performance against that reported in the literature. We describe this alternative work and compare it to ours in Section 5.5.

5.2 Extensions of AST Classifier Parameters

In the previous chapter (Chapter 4) we developed and investigated some classifier configurations and parameters to see the effectiveness of each and found that although differences between the configurations was marginal, that the root significance function performed very well in a number of settings. However, we have not investigated all the configurations we introduced in Chapter 3, in particular, word-based ASTs and the probabilistic significance function which we developed in Section 3.1.2; the investigation of these will be the two main thrusts of this chapter.

Hence, in Section 5.2.2, we retain the root significance function – as the best performing function from the previous chapter – and compare its performance against the probabilistic significance function. And in Section 5.2.3, we investigate the difference in performance that results from character-level versus word-level modelling of the classes.

However, we begin by considering effective and appropriate decision mechanisms that may be used to map individual class scores onto a set of predicted classes; we discuss this in the next section.

5.2.1 Decision Mechanisms For the Multi-Class Case

In the previous chapter, we dealt with a two class case. We thereby had a straight forward decision mechanism based on the ratio of the scores for each of the two classes, and hence, we simply used the mechanism described in Section 3.2, Equation 3.17.

In the multi-class case, if each document is associated with exactly one class, we may simply rank the classes by their score and assign the document to the highest ranking class. However, in many text classification tasks, and in particularly, our current case, a document may belong to one or more classes. We can of course mimic the two class case by taking for each class the ratio (as given in Equation 3.19) of the document score relative to the class against the document score relative to all other classes combined; thereby asking of each class whether or not the document belongs to it. Such an approach effectively turns the multi-class decision into a series of binary-class decisions; a technique that is widely used in the literature and often referred to as a *two-way ensemble* or the *one-vs-rest* technique. But this technique still leaves us the sometimes difficult task of either establishing a decision threshold, λ_{c_i}, for each of the classes, c_i, or using a single threshold for all classes and carefully normalising the scores for each document so that they fall consistently and reliably within a given range.

We experimented with both of these techniques, but also developed and tested another method that allowed us to mainly avoid the concerns over normalisation and class-specific thresholds. Under this method, we score all classes according the OMS function, then cluster them, by their OMS value, into positive and negative clusters, taking the former as the set of predicted classes. A single *cluster bias* is used to adjusted and optimise the overall accuracy of the classifier, such that the distance, d, between a class, c_i, and the positive cluster, C^+, is calculated as:

$$d(c_i, C^+) = bias * (centroid(C^+) - score(c_i)) \tag{5.1}$$

The clustering algorithm we use is generally referred to as k-means clustering [40] – see Algorithm 5.1 below. In this case, we set $k = 2$, meaning that we divide the classes into two groups: positive and negative. The positive group initially contains only the highest scoring individual class, with all others placed in the negative group.

5.1: K-Means Clustering Algorithm

```
Step 1. Begin with a decision on the value of k = number of clusters; in our current case, k = 2.
Step 2. Divide the individuals into k partitions - the division may be done by random or
        systematically, according to some rule(s). In our case, we form the positive partition (group)
        using only the most highly scoring class; with the remaining classes forming the negative
        partition (group).
Step 3. For each partition calculate its centroid (or mean).
Step 4. Take each class in sequence and compute its distance from the centroid of each of the
        clusters (partitions). If a class is not currently in the cluster with the closest centroid, switch it to
        the closer cluster and update the centroid of both clusters.
Step 5. Repeat step 3 until convergence is achieved; that is until an entire pass through the classes
        causes no new assignments.
```

We show in Section 5.4 that the clustering of classes by scores is highly effective and makes the classifier's performance very stable under several different classification parameter settings.

5.2.2 Choice of Significance Functions

In the previous chapter (Chapter 4) we experimented with a number of simple definitions of the significance function and two methods of match normalisation. We found that there was little to separate the performance, but some gains were achieved by using the root significance function and match permutation normalisation. In this chapter, we will concern ourselves less with normalisation and focus more on only two approaches to scoring: we retain the root significance function and compare its performance under various conditions against the probabilistic significance function, as described in Section 3.1.2 (Equation 3.15).

When using the root significance function, we use a ratio measure whereby we calculate the score for the class, $OMS(Q|T_c)$, and the negative class conditional score, $OMS(Q|T_{\bar{c}})$, and take the ratio of the former over the latter, as described in Section 3.2, Equation 3.19; which is equivalent to the ratio we took in Chapter 4, but is now applied in a multi-class setting. By taking such a ratio for each class, we introduce a measure of the corpus frequencies, which are otherwise not addressed in the root scoring function. In contrast, when using the probabilistic significance function, which directly addresses corpus frequencies, we calculate only the class score, $OMS(Q|T_c)$.

5.2.3 Character-Level vs Word-Level Classification

One of the aims of this chapter is to compare the performances of the word-based AST classifier and character-based AST classifier. In the previous chapter we concentrated our attention on the character-based AST classifier, arguing that it afforded a special advantage when handling the character-level obfuscation employed in many spam messages; and the results we obtained supported this view. But such an advantage does not exist in a more general text classification context, and, as we have said, particularly so in the case of the Reuters data set.

In general text classification, the classifier will generally not encounter deliberate obfuscations, but there may still be other advantages to character-based modelling of the text. For example, character-level modelling can capture information about sub-word features such as punctuation, word-boundaries, text formatting, or even domain specific meta-textual features such as HTML tags on web pages. All such features can give clues to the class of a document, and some are relevant to our current target data set, as we saw by comparing the stories shown in Figure 5.2 and 5.1.

The classification methods we developed in Chapters 2 and 3 are agnostic as to whether we use characters or words as the basic units for classification features. And as the Reuters data set is widely considered to lend itself well to word-based classification, it makes sense to experiment with word-based AST classifiers and compare their performance against character-based AST classifiers.

Furthermore, it is certainly the case that context can be important in word-based classification, just as it is crucial in character-based ones. Using word-based n-gram models allows the classifier to distinguish between word features that might occur is several classes. For example, the term “stock” may occur in the ‘grain’ class, and in the ‘acq’ class, but in the former it is more likely to occur as “grain stock” and in the latter as “stock market”.

In both the word- and character-based models, we use exactly the same method to evaluate the OMS, except that in the former we perform some preprocessing that removes punctuation. For the character-level classier, as before, we perform no preprocessing of the text, but add the title and body of each news story *as is*. In short, we argue that although this has the disadvantage of inserting a great deal of noise into the model, it has the advantage of retaining a great deal of useful information such as the order of terms and the inflection of words; as well as numerals and formatting.

5.2.4 Tree Depth

Utilising a different term base for our suffix trees raises again the question of depth - for both the character- and the word- based AST classifiers (ASTC).

In the case of the character-based ASTC, we examined, in Chapter 4, Section 4.4.1, the effect of varying depth on the performance and found that it benefited from greater depth, though by a diminishing marginal amount. We settled on using a character depth of 8, which is important when we are dealing with spam classification because a greater depth is useful when we are trying to jump over the obfuscations that are common in spam messages; however, a lower depth may well be adequate in the case of the Reuters collection, where there are no such deliberate obfuscations (though there may be, and indeed are, numerous spelling mistakes).

In the case of the word-based ASTC, we have so far not examined the effect of depth variation. At a word depth of 1, the methods we have developed approximate a basic TF-IDF or naive Bayes algorithm based on the vector space model and the term independence (‘bag-of-words’) assumption. Increasing the depth beyond 1 introduces context sensitivity and thereby increases the power of the classifier, but it unlikely we will need a depth as great as that for a character-based classifier.

5.3 Experimental Setup

All the experiments described in this chapter were conducted on a subset of the Reuters 21578 data set under the ModApte split; details of the subset we use are given in Section 5.3.1 below. No cross-validation experiments were done because the data set is divided into standardised training and testing sets (see Section 5.1).

Performance was assessed using essentially similar measures to those introduced in the previous chapter (Chapter 4), but suitably adapted to apply in the multi-class context. We take a closer look at these performance measures for the multi-class case in Section 5.3.2.

5.3.1 Reuters Subset

How researchers use the Reuters data set can vary considerably. Some researchers [15] run experiments on all 118 categories, but quote results from only the largest 10, while others [63] may use only the 12

Class	Total Training	Total Testing
Earn	2877	1087
Acq	1650	719
Money-fx	538	179
Grain	433	149
Crude	389	189
Trade	369	117
Interest	347	131
Ship	197	89
Wheat	212	71
Corn	181	56

Table 5.1: The largest 10 classes in the Reuters-21578 Collection.

largest (most frequent) only. However, most often, researchers seem to prefer to concentrate on the 10 largest classes [6, 36, 44, 54]. We follow the latter and use the largest 10 classes, shown in Table 5.1. Together, these classes account for more than 75% of all the documents in the Reuters-21578 collection. Table 5.1 shows the number of training and testing examples in each of the 10 largest classes arranged in order of size.

Of course, which approach is used can undoubtedly have an effect on performance [12] and so in some comparisons it should be kept in mind that it is not possible to draw hard conclusions.

5.3.2 Performance Measures

We mainly follow the literature [15, 13, 63, 13] and use the decision measures of Precision (Pr), Recall (Re), the F1 measure which combines the two, and the 'breakeven' point where precision equals recall. Additionally, when dealing with multiple classes, is it often useful to consider an average over all the classes. This is typically done in two ways: by *microaveraging* and by *macroaveraging*. Recollect from Section 4.4.3 the terms: true positives (TP), false positives (FP) and false negatives (FN). The meaning in the multi-class case are very similar, but we think in terms of a document belonging either to a class c or to $\overline{c}$. Hence, a TP is a document, d^c, belonging to c which is correctly classified as c, a FN is d^c incorrectly classified as $\overline{c}$ and a FP is a document, $d^{\overline{c}}$, belonging to $\overline{c}$, incorrectly classified as c. We then define the two methods of averaging as follows:

$$MicroAveragedPrecision = mPr = \frac{\sum_{c \in C} TP_c}{\sum_{c \in C}(TP_c + FP_c)} \tag{5.2}$$

$$MicroAveragedRecall = mRe = \frac{\sum_{c \in C} TP_c}{\sum_{c \in C}(TP_c + FN_c)} \tag{5.3}$$

$$MacroAveragedPrecisions = MPr = \frac{1}{C}(\sum_{c \in C} Pr_c) \tag{5.4}$$

$$MacroAveragedRecall = MRe = \frac{1}{C}(\sum_{c \in C} Re_c) \tag{5.5}$$

And from these definitions, we can define macro- and micro-F1 using either macro- or micro-averaged precision and recall as before:

$$F1 = \frac{2(Pr * Re)}{Pr + Re} \tag{5.6}$$

In the two-class case of Chapter 4, we additionally made extensive use of ROC graphs (as discussed in Section 4.4.3) to demonstrate the performance of the classifier over a range of thresholds, but such graphs are difficult (though possible [27]) to define in a multi-class case and so we use instead, precision-recall (PR) curves [47, 35]. The two curves are closely related [10] and give similar insights into the performance of the classifier under varying conditions, but curve in opposite directions such that an optimal classifier would achieve a point on a PR curve at the top right of the graph space, whereas for an ROC curve it would achieve a point on the top left.

5.4 Results

The classifier parameters allow for a large number of possible permutations, but we avoid presenting results from each of these. Instead, we begin, in Section 5.4.1 by using only the *character-based AST classifier* (ASTC-C) to show that the score clustering decision mechanism exhibits a distinct advantage. We obtained similar results for the *word-based AST classifier* (ASTC-W), but do not present them here.

We then examine the effect of depth on the ASTC-C in Section 5.4.2 and establish an effective depth

Class	th	Pr (%)	Re (%)	F1 (%)
earn	1.10	94.80	95.68	95.24
acq	1.09	93.94	96.94	95.41
money-fx	1.09	63.30	94.41	75.78
grain	1.08	67.91	85.23	75.60
crude	1.09	73.30	85.71	79.02
trade	1.09	53.67	81.20	64.63
interest	1.11	58.33	69.47	63.41
wheat	1.09	32.80	85.92	47.47
ship	1.10	51.18	73.03	60.19
corn	1.11	29.79	50.00	37.33
Macro-Av.	n/a	**61.90**	**81.76**	**70.46**
Micro-Av.	n/a	**77.90**	**90.96**	**83.93**

Table 5.2: Precision (Pr), Recall (Re) and F1 rates for an ASTC of depth 8 using a root significance function (rASTC-C8) and individual class-thresholds (th).

to use in the context of more general text classification tasks. Having established an effective depth for the ASTC-C, we move on to use it as a comparator for studying the performance of the ASTC-W in Section 5.4.3.

Comparisons between the root and probabilistic significance functions are made throughout.

5.4.1 Decision Mechanisms for Predicting Multiple Classes

When predicting several possible classes, a typical decision mechanism that is used in text classification and machine learning is to introduce a *threshold* score for each class. As before, the query is scored against each class, but this time we select all the classes which gain scores above their individual threshold. The threshold may be established for each class in a post-training phase using a sample of the training data, but they may also be adjusted by users of the system.

Table 5.2 presents the results for a character-based ASTC (ASTC-C) of depth 8 using the **root** significance function and the individual class thresholds which produce the highest F1 accuracy. From a classification accuracy point of view, the results are competitive with some existing methods but fall below the results of others [63, 15]. The largest classes, ‘earn’ and ‘acq’, have high precision and recall, but the smallest classes tend to have very low precision.

The performance of the probabilistic significance function under otherwise identical parameter settings is much worse. Table 5.3 shows the results from a character-based ASTC of depth 8 using the

Class	th	Pr (%)	Re (%)	F1 (%)
acq	0.00	30.55	99.86	46.79
corn	0.26	2.20	100.00	4.31
crude	0.00	10.82	100.00	19.53
earn	0.00	43.46	100.00	60.59
grain	0.00	11.84	77.85	20.55
interest	0.00	9.39	93.13	17.06
money-fx	0.00	10.45	95.53	18.83
ship	0.00	7.50	96.63	13.93
trade	0.00	6.73	99.15	12.61
wheat	0.22	2.79	100.00	5.43
Macro-Av	n/a	**13.57**	**96.21**	**23.79**
Micro-Av	n/a	**14.79**	**98.03**	**25.70**

Table 5.3: Precision (Pr), Recall (Re) and F1 rates for classes using a character-based AST classifier of depth 9 (pASTC-C8) using probabilistic significance function (pSF) with individual thresholds (th) on the largest 10 classes in the Reuters-21578 Collection.

probabilistic significance function (pASTC) and the individual thresholds which provide the highest F1 accuracy. Recall is generally very high, but this corresponds with a very low precision. We look at the reasons for this in the next section.

Predicting multiple classes using clustering

One of the difficulties with a single threshold for each class is that in practice, the range of scores that one query receives over all the classes may differ considerably from the range of scores achieved by another query; so much so that the lowest class score for one document may be higher than the highest class score for another document. Such variation makes it impossible to find a single class threshold which will accurately identify all true positive classes. Scores are, of course, normalised (at the very least by the length of the query) to maintain some consistency over different queries, but this is not always effective enough.

An alternative approach is to look at the distribution of scores and group the classes into positive and negative clusters regardless of the absolute score for each. To illustrate why such an approach is effective consider the examples shown in Figures 5.4, 5.3 and 5.5; each shows a numbered news story from the test set of the ModApt split. Each example belongs to the ‘acq’ class and the third belongs additionally to the ‘crude’ class. The range of scores for story 15542 (Figure 5.3) is from 0.94 (for the ‘trade’ class) to 1.65 (for the ‘acq’ class); while the range of scores for story 15161 (Figures 5.4) is from

1.03 (for the ‘trade’ class) to 1.09 (for the ‘aqc’ class). The ‘acq’ class attains a lower score for the latter story than the ‘corn’, ‘crude’, ‘earn’, ‘grain’, ‘interest’ and ‘wheat’ classes attain for the former. Hence any ‘acq’ class threshold that includes both these articles as true positives would also include all these other classes as false positives. As such, it would be impossible to use a threshold for the ‘acq’ class which would correctly classify both these documents.

If, instead, we consider the distribution of the scores within their own range, we see that the true classes are usually distinct from the other classes. This is shown in the charts that accompany each story in Figures 5.4, 5.3 and 5.5. In the first example, the true class (‘acq’) scores over 1.6, well above all the other classes which are grouped around a score of 1.1. This is quite a high range of scores for all the classes, which is arguably due to the ’story’ being only a headline: journalistic headlines tend to concentrate many significance terms into short term-sequences; and hence the length normalised scores tend to be high. The second example shows a complete news story where the range of scores is much lower. The true class, ‘acq’, now scores less than 1.1, but is still clearly separated from all the other classes whose scores are grouped around 1.04. The final example shows a story belonging to the ‘acq’ and ‘crude’ classes, which are now grouped together around a score of 1.1, while all other classes are grouped around the much lower 1.025 mark.

Examining such charts strongly suggests the method of clustering classes based on their scores and such an approach was used to obtain the results shown in Table 5.4, which proved to be the most successful of those we tested. The table shows the results at the bias which results in the highest F1 accuracy. The results show a marked improvement in all regards, with no anomalously low performances and a mean performance of 83.59% for all measures. This performance can be improved by application of the ik-means algorithm [41], which fixes the positive and negative cluster means; the approach raises the F1 score to 84.85%. The results using fixed positive and negative cluster means is shown in Table 5.5. The approach of fixing positive and negative cluster means has the significant additional advantage of faster execution times because recalculation of the mean is not necessary and the clustering algorithm will complete in linear (to the number of classes) time .

The argument so far has taken examples of documents scored using the additive root significance function (rSF), but a similar argument applies to the probabilistic significance function (pSF), which actually shows a much more dramatic improvement, indicating that the variation in the range of scores

```
<REUTERS NEWID=15542> <TOPICS>
 <D>acq</D>
</TOPICS>

<Title>E.F. HUTTON LBO INC SAID TENDER OFFER BY PC ACQUISITTION
FOR PUROLATOR COURIER EXPIRED </Title>

<Body />
```

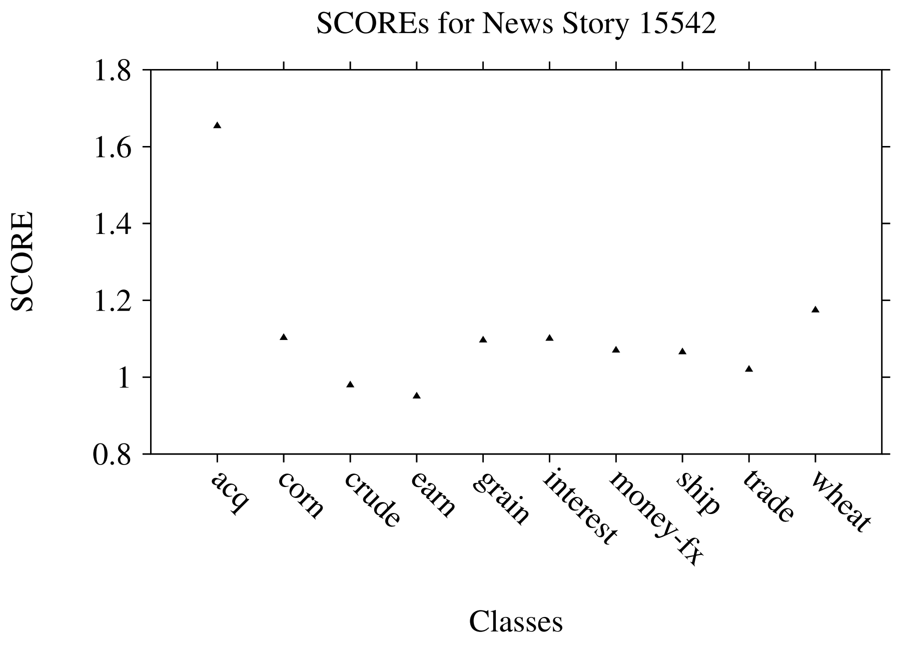


Figure 5.3: Example Reuters News Story (top) belonging to the ‘acq’ class; and the scores against each of the largest 10 classes (bottom)

across documents is far greater than that for the root significance function (rSF). The results for variable and fixed positive and negative cluster means for the probabilistic significance function (pSF) are shown in Tables 5.6 and 5.7.

The probabilistic significance function shows a small improvement from ik-mean clustering, as can be seen by comparing the macro- and micro averages in Tables 5.6 and 5.7, but, as with the rSF, the difference is small.

Moreover, there seems to be little to separate the two scoring functions, though the rSF achieves a marginally better performance. However, this is only at one particular cluster bias. If we look at

```
<REUTERS NEWID=15161>
 <TOPICS>
    <D>acq</D>
</TOPICS>

<Title>U.K. GEC DECLINES COMMENT ON U.S. PURCHASE RUMOUR</Title>

<Body>General Electric Co Plc <GECL.L> (GEC) declined comment on
 rumours on the London stock market that it is planning another
 purchase in the U.S. Medical equipment field, in addition to its
 existing U.S. Subsidiary <Picker International Inc>.
    A GEC spokesman said that it is company policy not to comment
 on acquisition rumours.
    Stock Exchange traders said the rumour helped GEC's share
 price to rise 5p, to a final 206p from yesterday's closing price
 of 201p. </Body>
```

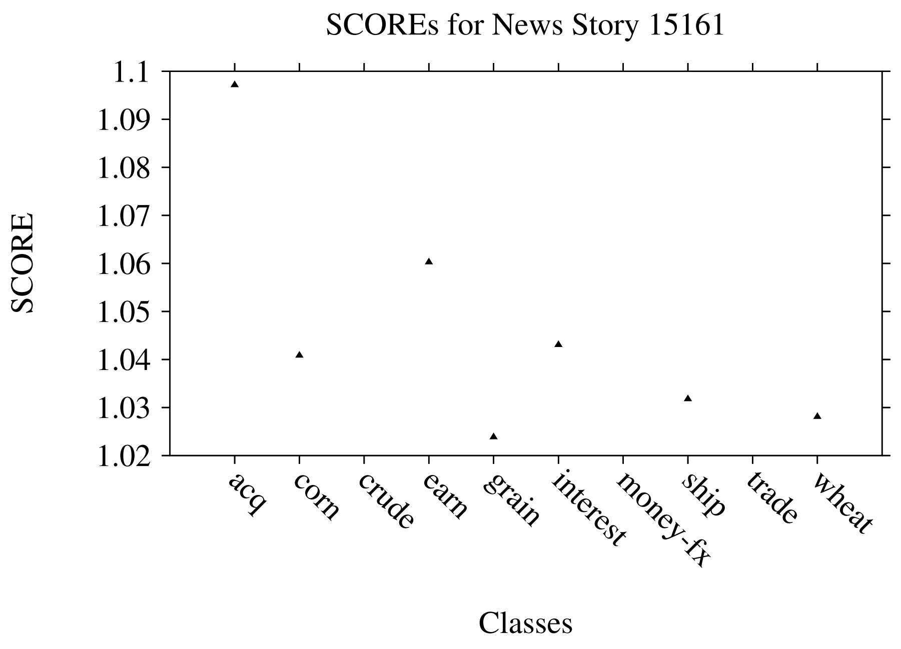


Figure 5.4: Example Reuters News Story (top) belonging to the 'acq' class; and the scores against each of the largest 10 classes (bottom)

```
<REUTERS NEWID=16007> <TOPICS>
 <D>acq</D>
 <D>crude</D>
</TOPICS>

<Title>NERCI <NER> UNIT CLOSES OIL/GAS ACQUISITION</Title>

<Body> Nerco Inc said its oil and gas unit closed the acquisition
 of a 47 pct working interest in the Broussard oil and gas field
 from <Davis Oil Co> for about 22.5 mln dlrs in cash.
    Nerco said it estimates the field's total proved developed and
 undeveloped reserves at 24 billion cubic feet, or equivalent, of
 natural gas, which more than doubles the company's previous
 reserves.
    The field is located in southern Louisiana.
</Body>
```

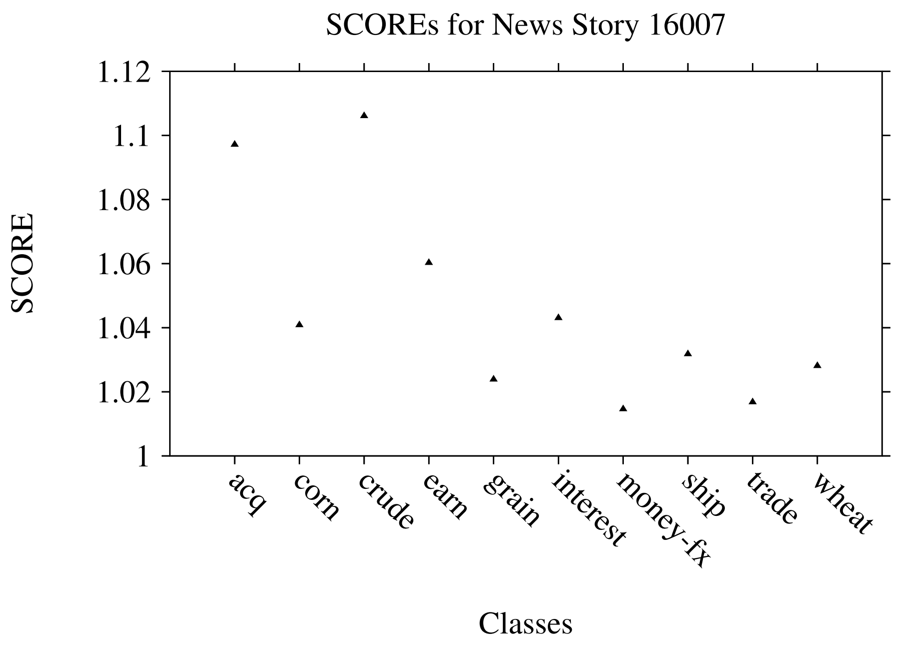


Figure 5.5: Example Reuters News Story (top) belonging to the 'acq' and 'crude' class; and the scores against each of the largest 10 classes (bottom)

Class	bias	Pr (%)	Re (%)	F1 (%)
acq	4.11	96.00	97.18	96.59
corn	4.11	59.26	85.71	70.07
crude	4.11	93.17	79.37	85.71
earn	4.11	97.28	98.53	97.90
grain	4.11	98.80	55.03	70.69
interest	4.11	83.46	80.92	82.17
money-fx	4.11	82.81	88.83	85.71
ship	4.11	85.19	76.67	80.70
trade	4.11	73.65	92.37	81.95
wheat	4.11	65.91	81.69	72.96
Macro-Av	n/a	**83.55**	**83.63**	**83.59**
Micro-Av	n/a	**91.59**	**91.65**	**91.62**

Table 5.4: Precision (Pr), Recall (Re) and F1 rates for classes using a character-based ASTC of depth 8, the **root** significance function (rSF) and the clustering approach for class allocation on the largest 10 classes in the Reuters-21578 Collection.

Class	bias	Pr (%)	Re (%)	F1 (%)
acq	3.43	95.35	97.35	96.34
corn	3.43	54.55	87.50	67.20
crude	3.43	90.91	82.42	86.46
earn	3.43	97.11	98.99	98.04
grain	3.43	99.04	74.10	84.77
interest	3.43	85.04	83.72	84.38
money-fx	3.43	80.98	89.76	85.14
ship	3.43	84.42	77.38	80.75
trade	3.43	72.34	91.89	80.95
wheat	3.43	67.44	87.88	76.32
Macro-Av	n/a	**82.72**	**87.10**	**84.85**
Micro-Av	n/a	**91.07**	**93.47**	**92.25**

Table 5.5: Precision (Pr), Recall (Re) and F1 rates for classes using a character-based ASTC of depth 8, the **root** significance function (rSF) and the clustering selection approach using fixed positive and negative cluster means on the largest 10 classes in the Reuters-21578 Collection.

the performance of each of these significance function at varying cluster bias values, there are further differences in their behaviour.

Figure 5.6 compares the micro-averaged precision vs micro-averaged recall performance of the root and probabilistic (prob) significance functions for a range of cluster bias values. The probabilistic significance function (pSF) now appears to clearly outperform the root significance function (rSF) for most cluster bias values and has a much more evenly convex curve - indicating that it is a more stable

ClassPath	th	Pr	Re	F1
acq	3.56	95.52	97.77	96.63
corn	3.56	50.49	92.86	65.41
crude	3.56	83.70	81.48	82.57
earn	3.56	99.06	97.33	98.19
grain	3.56	91.27	77.18	83.64
interest	3.56	75.82	88.55	81.69
money-fx	3.56	82.47	89.39	85.79
ship	3.56	71.68	91.01	80.20
trade	3.56	72.55	94.07	81.92
wheat	3.56	57.52	91.55	70.65
Macro-Av	n/a	**78.01**	**90.12**	**83.63**
Micro-Av	n/a	**88.85**	**93.79**	**91.26**

Table 5.6: micro-Precision (Pr), micro-Recall (Re) and F1 rates for classes using a character-based ASTC of depth 8, the **probabilistic** significance function using clustering approach for class allocation on the largest 10 classes in the Reuters-21578 Collection.

ClassPath	th	Pr	Re	F1
acq	3.02	95.79	97.55	96.66
corn	3.02	57.47	89.29	69.93
crude	3.02	90.48	80.42	85.15
earn	3.02	96.76	98.90	97.82
grain	3.02	98.99	65.77	79.03
interest	3.02	80.88	83.97	82.40
money-fx	3.02	80.50	89.94	84.96
ship	3.02	81.40	77.78	79.55
trade	3.02	71.71	92.37	80.74
wheat	3.02	66.33	91.55	76.92
Macro-Av	n/a	**82.03**	**86.75**	**84.33**
Micro-Av	n/a	**90.50**	**93.07**	**91.77**

Table 5.7: micro-Precision (Pr), micro-Recall (Re) and F1 rates for classes using a character-based ASTC of depth 8, the **probabilistic** significance function using clustering selection approach with **fixed** positive and negative means on the largest 10 classes in the Reuters-21578 Collection.

classifier which has made a better separation between the classes. However, towards the top left of the curves, the curve of the rSF rises above the pSF curve and it is here that the F1 values are maximised for both, which is why the rSF appears to perform better if we simply examine the tables of results at F1-maximising bias values. All that said, it is also clear that there is very little overall difference in the performance of each significance function.

Low cluster bias values yield the points at the bottom right of the curves - which are associated with

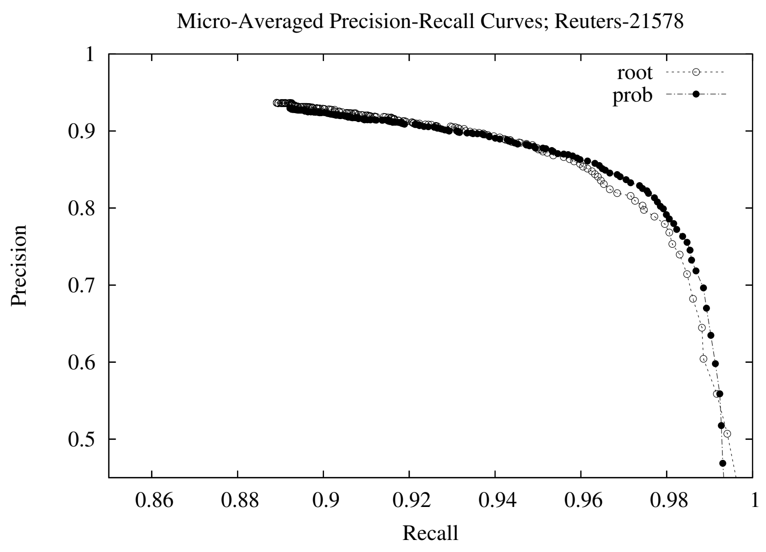


Figure 5.6: Micro-Averaged Recall against Micro-Averaged Precision results on the Reuters-21578 data set using the root and probabilistic (prob) significance functions.

a high recall and low precision. This is because low bias values make it easier for classes to be placed into the positive cluster, as can be seen by considering Equation 5.1. A bias of 1 achieves a recall of approximately 98%, and so is represented on the graph by the second (rSF) or third (pSF) point on their respective curves. The marginal improvement with each increment of the bias value is initially large (toward the bottom right of the curves), reflected in the curves by well spaced points, but eventually, the marginal improvement diminishes until there is almost no change in performance from each increment of the bias - this is reflected in the curves by a concentration of the points at the top right of the graph. This observation is useful in the practical use of the classifier because it means we can safely utilise a high bias that will ensure a position on the curve where the performance has stabilised, without worrying too much about ‘over-shooting’ the maximal performance level.

A similar story may be seen in the macro-averaged precision-recall curves shown in Figure 5.7. Here there is a more demarcated crossing of the two curves representing the performance of rSF and pSF. Again, the highest F1 accuracy is reached at the top left of the graph where the rSF curve rises above pSF curve, but that the latter offers a more stable classifier with a better separation between the classes can be seen clearly from the more evenly concave curve.

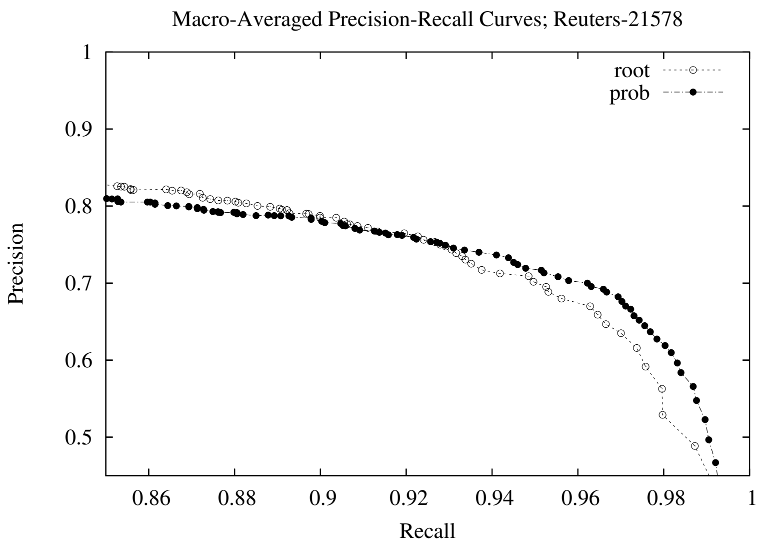


Figure 5.7: Macro-Averaged Recall against Micro-Averaged Precision results on the Reuters-21578 data set using the root and probabilistic (prob) significance functions.

5.4.2 Effect of Depth Variation on Character-Based ASTCs

All the analysis so far has been done using character-based annotated suffix tree classifiers (ASTC-Cs) of depth 8, which is a depth we established as effective in the domain of spam filtering (Chapter 4). However, as we have said, such a large depth may not be necessary in a broader text classification domain. Hence, let us now consider how performance changes if we reduce the depth of the tree.

Figure 5.8 shows the micro-averaged precision-recall curves for a ASTC-C of depths 2 to 8 using the rSF. Clearly, by the time we have reached a depth of 4, there is little to separate the performances of greater depths (i.e. from depth of 4 to 8). There is a little more separation between the depths if we consider the macro-averaged precision-recall curves for the same set of classifiers, as shown in Figure 5.9. Here we can see that at a depth of 8 (shown by light coloured triangles), the classifier performance has already begun to decline. We can see that although the differences are small, the classifier performance seems to peak at a depth of 5 (shown by dark squares). But separating the curves is only possible for lower cluster bias values i.e. towards the bottom right of the curves. Towards the top left of the curves (corresponding with a high cluster bias), curves for depths greater than 4 tend to merge. This region is also where the classifiers tend to achieve the highest F1 accuracy.

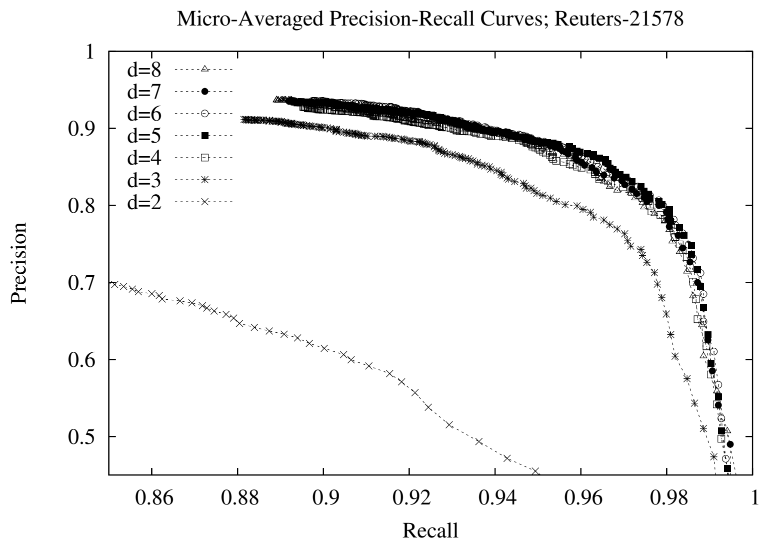


Figure 5.8: Micro-Averaged Precision-Recall curves for a character-based annotated suffix tree classifier (ASTC-C) using a root significance function (rSF) for tree depths 2 to 8.

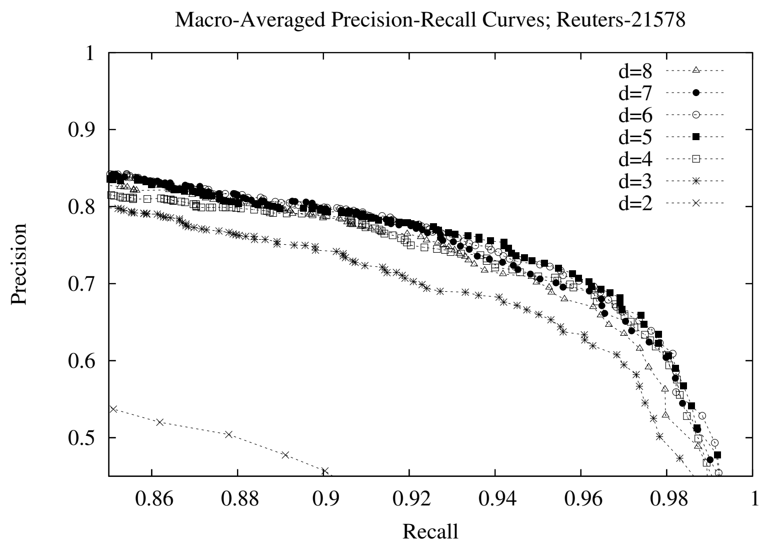


Figure 5.9: Macro-Averaged Precision-Recall curves for a character-based annotated suffix tree classifier (ASTC-C) using a root significance function (rSF) for tree depths 2 to 8.

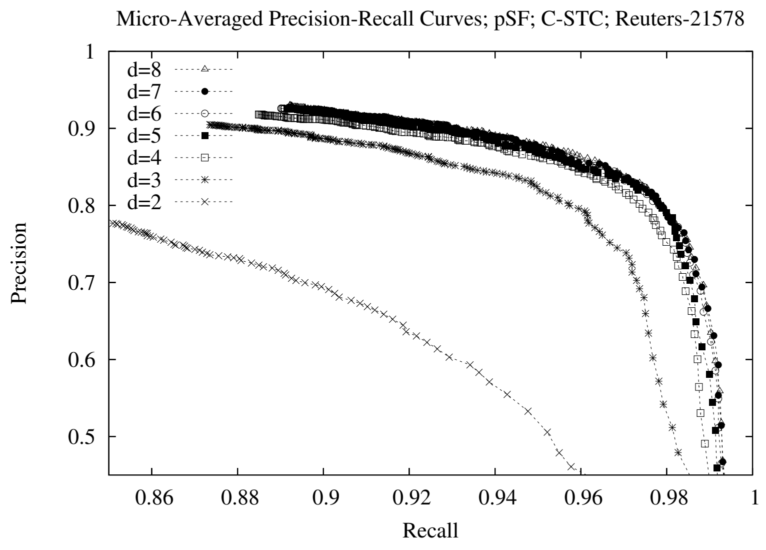


Figure 5.10: Micro-Averaged Precision-Recall curves for a character-based annotated suffix tree classifier (ASTC-C) using a probabilistic significance function (pSF) for tree depths 2 to 8.

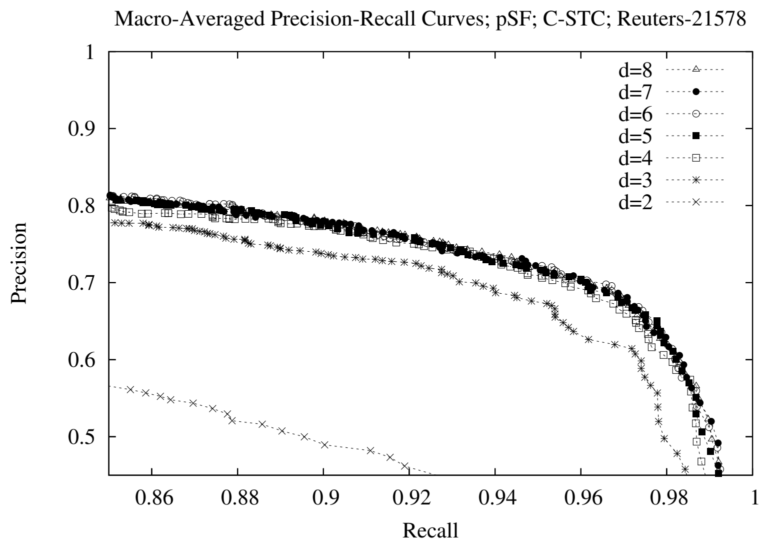


Figure 5.11: Macro-Averaged Precision-Recall curves for a character-based annotated suffix tree classifier (ASTC-C) using a probabilistic significance function (pSF) for tree depths 2 to 8.

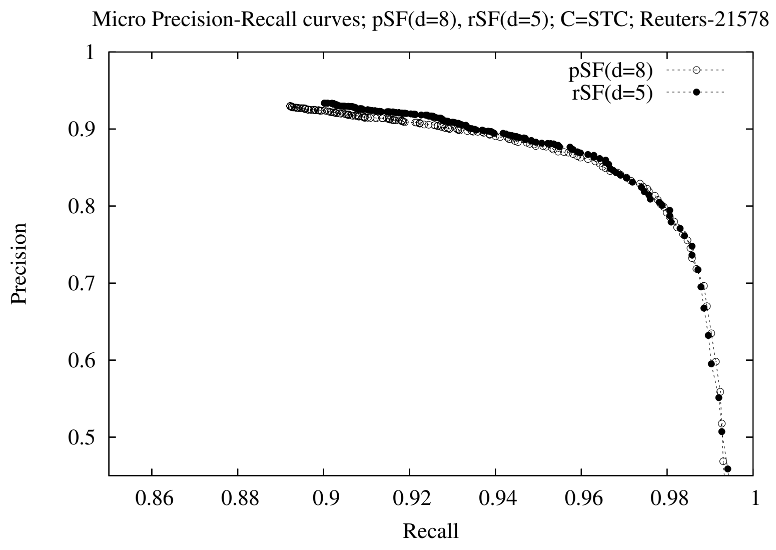


Figure 5.12: Micro-Averaged Precision-Recall curves comparing an rSF classifier of depth 5 with a pSF classifier of depth 8.

In the case of the C-ASTC using the pSF, there is a much more uniform performance for the different depths. Figure 5.10 shows the micro-averaged precision-recall curves and Figure 5.10 shows the macro-averaged curves. For depths of 5 and above, there are no discernable differences in performance, but interestingly, there is also no decline in performance and the classifier's performance seems to stablise.

Notice that this means that the comparisons made in Figures 5.6 and 5.7 are not fair to the rSF based classifer because they compare performance at a depth of 8, whereas the rSF achieves its best performance at depth 5. Figures 5.12 shows that at a depth of 5, an rSF classifier matches the performance of a pSF classifier of depth 8 in terms of micro-averaged F1 accuracy; and Figure 5.13 shows that an rSF classifier of depth 5 can very closely match the performance of an rSF classifier of depth 8 in terms of macro-averaged F1 accuracy.

Overall, we can interpret these graphs as indicating that less tree depth is needed for data sets such as Reuters compared to the spam data sets we used in Chapter 4. This is very understandable because, as we have said, the Reuters data does not contain any of the deliberate obfuscations such as those which generally characterise spam email.

We can also see that the choice of significance function does not seem to be crucial to performance; differences may be smoothed out by altering the depth or cluster bias.

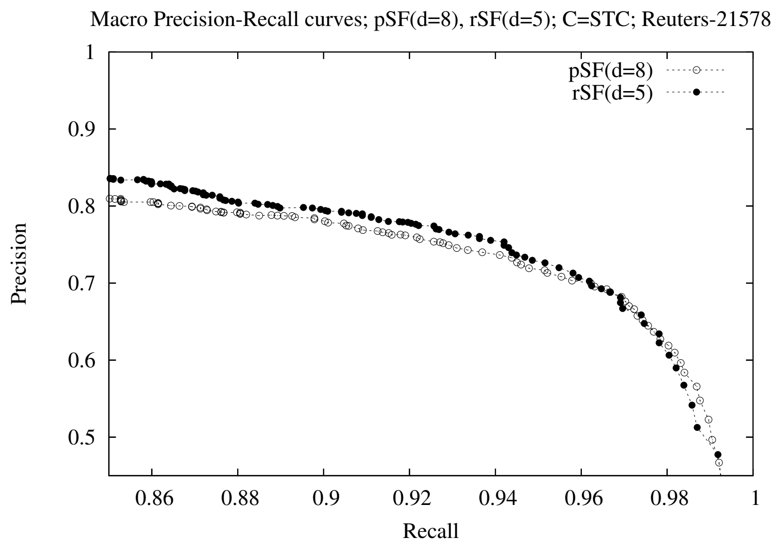


Figure 5.13: Macro-Averaged Precision-Recall curves comparing an rSF classifier of depth 5 with a pSF classifier of depth 8.

5.4.3 Character-Based vs Word-Based Classification

In contrast to the character-based classifier, we found that in the case of word-based classifiers, the choice of significance function and depth can have a large effect on performance.

We begin by considering word- and character-based classifiers of the same depth - specifically, a depth of 8. However, such a high depth is unlikely to be necessary, so in Section 5.4.3, we look at the effect of altering depth.

Figure 5.14 shows that, surprisingly, the rSF performance drops dramatically when applied to a word-based AST classifier: both precision and recall are very low and there is a linear trade-off between them. The results are very different with a pSF, for which performance improves when applied to a word-based ASTC, as shown in Fig 5.15. As we would expect for the Reuters data set, the word-based classifier outperforms the character-based classifier at almost all cluster bias values.

Why the root significance function (rSF) performs so badly on word-based ASTCs is not clear, but we can make some observations. As we argued in Chapter 4, the advantage of the rSF is that it is more sensitive to low probabilities (more sensitive than a linear significance function would be). This would make it better at distinguishing classes which are separable on the basis of many features which have

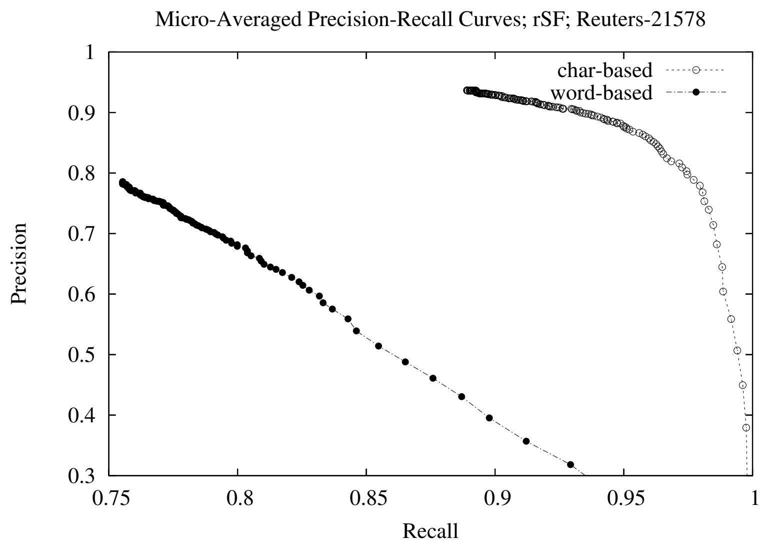


Figure 5.14: Micro-Averaged Recall against Micro-Averaged Precision results on the Reuters-21578 data set using the root significance function (rSF) on word-based and character-based classifiers.

low frequencies, as in the case of the spam filtering domain. But Reuters is characterised by classes which are highly separable based on a few frequently occurring terms.

It is also the case that the probabilistic significance function (pSF) takes account of the relative frequency of terms in the same way that TF-IDF methods do - as we showed in Section 3.1.2. The rSF method only takes account of relative frequencies across classes in an indirect way through the score ratio which is used in the decision process, but this ratio is taken only after the whole document has been evaluated and not on a term-by-term basis.

However, these suggestions are fairly speculative and further investigation would be needed to establish exactly why the rSF performs so badly when used with a word-based ASTC (ASTC-W). For now, let us conclude only that this experiment demonstrates that use of the probabilistic significance function seems to lead to a more robust classifier.

Effect of Depth Variation on Word-Based ASTCs

Experiments varying the depth of the profiles trees also had surprising results. Figure 5.16 shows how the precision-recall curves for a word-based classifier of depths 1 to 8 (Figures (a) to (h)) using the

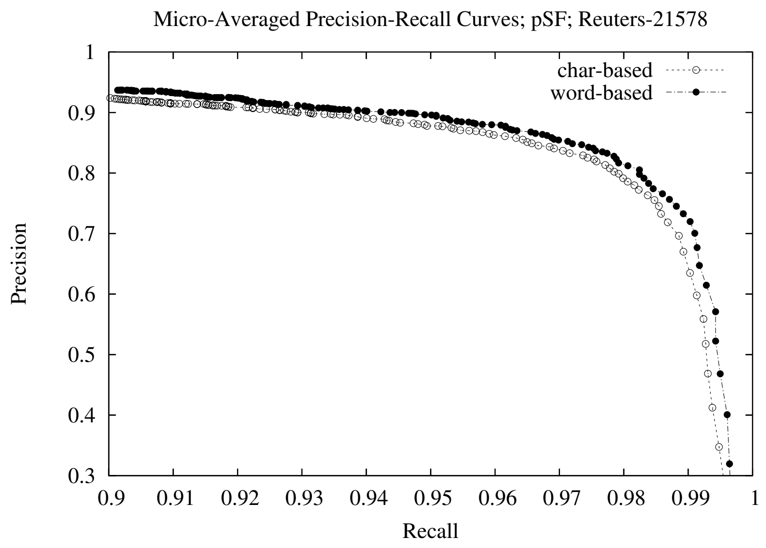
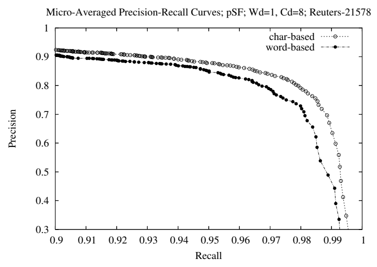


Figure 5.15: Micro-Averaged Recall against Micro-Averaged Precision results on the Reuters-21578 data set using the probabilistic significance function (pSF) on word-based and character-based classifiers.

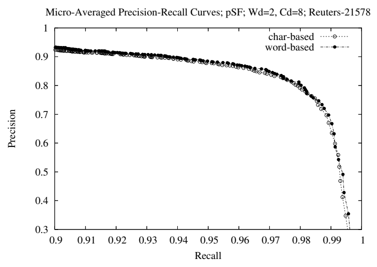
pSF change relative to a character-based classifier of depth 8, also using the pSF. We have already seen Figure 5.16h in the previous section, but now we can see how this curve emerges as the depth increases. At a depth of 1, the word-based classifier performs significantly worse than the character-based classifier at all cluster bias values. The PR curve of the ASTC-W is also not as smooth or uniformly concave as the ASTC-C, but the curves tend to move closer together towards the top left, which is a region of the graph that coincides with the maximum F1 scores for classifiers of both term-bases. And this is a characteristic of the two curves at for all word depths.

Increasing the ASTC-W depth to 2 (Figure (b)), removes the difference in performance almost completely and the two PR curves follow almost exactly the same line over the entire graph. Increasing the depth further leads to the ASTC-W marginally outperforming the ASTC-C, but the ASTC-W curve looses its smoothness and exhibits areas of local convexity. As we increase the word-depth further, the actual overall performance of the ASTC-W does not improve much at all, but its PR curve becomes far more smooth and more uniformly convex such that by the time we have increased the word-depth to 7, the PR curve for the ASTC-W traces a uniform line above the PR curve for the ASTC-C of depth 8.

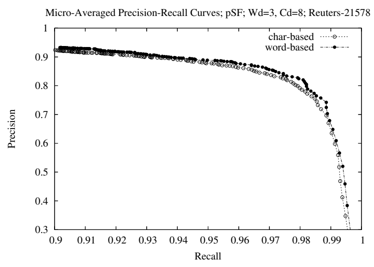
A very different picture is apparent in the case of the word-based classifier using the rSF: as depth



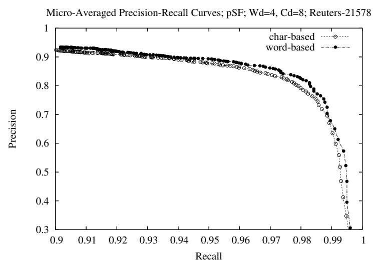
(a) Word Tree Depth (Wd) = 1



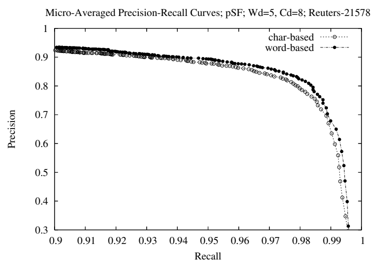
(b) Word Tree Depth (Wd) = 2



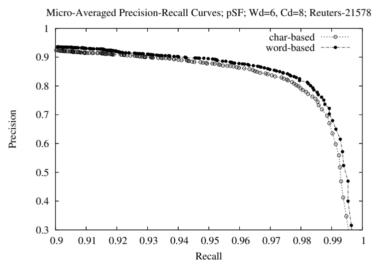
(c) Word Tree Depth (Wd) = 3



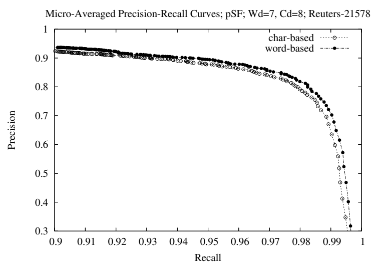
(d) Word Tree Depth (Wd) = 4



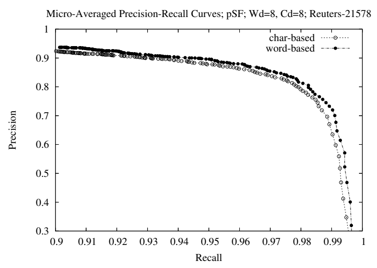
(e) Word Tree Depth (Wd) = 5



(f) Word Tree Depth (Wd) = 6

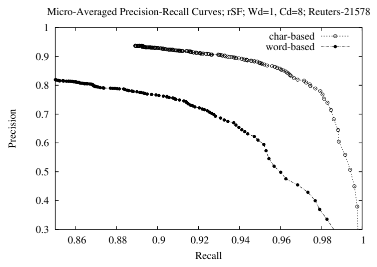


(g) Word Tree Depth (Wd) = 7

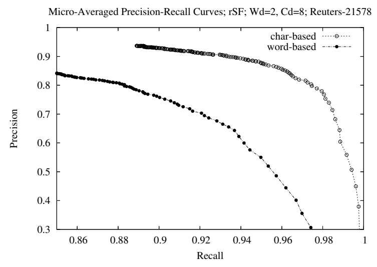


(h) Word Tree Depth (Wd) = 8

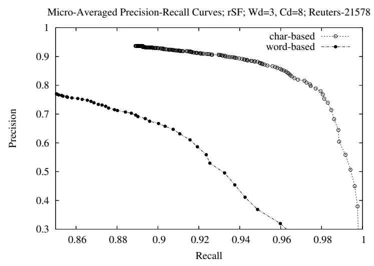
Figure 5.16: Word-based classifiers for depths 1-8 shown against a character-based classifier of depth 8 using a probabilistic significance function.



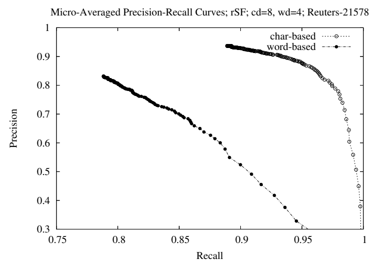
(a) Word Tree Depth (Wd) = 1



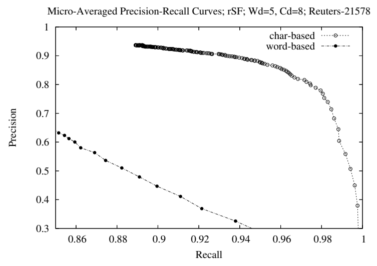
(b) Word Tree Depth (Wd) = 2



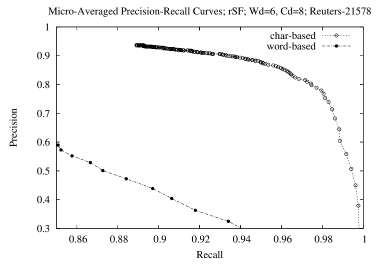
(c) Word Tree Depth (Wd) = 3



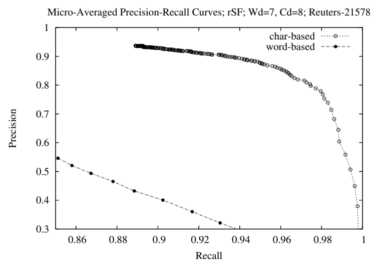
(d) Word Tree Depth (Wd) = 4



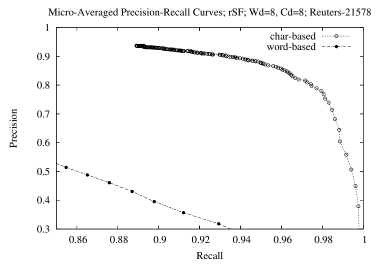
(e) Word Tree Depth (Wd) = 5



(f) Word Tree Depth (Wd) = 6



(g) Word Tree Depth (Wd) = 7



(h) Word Tree Depth (Wd) = 8

Figure 5.17: Word-based classifiers for depths 1-8 shown against a character-based classifier of depth 8 using a root significance function.

increases, the performance actually moves in the opposite direction. Indeed ASTC-W using a rSF performs best at a word depth of just one. Such a result is very hard to explain and further work is needed to investigate this matter.

5.5 Assessment: Related Work

Text classification in general and the Reuters data set in particular have been tackled by researchers using a variety of methods. The majority of methods are word-based [23, 15, 6], but some recent work has attempted to tackle the problem using character-based classifiers [33, 54].

Results in the literature are also reported in different ways - either using macro- and micro-F1 scores or using precision-recall break-even points. We calculated all these values from our experiments so that they were comparable with all those reported in the literature. The comparisons are made below against both word- and character-based approaches.

5.5.1 Word-Based Classifiers

Traditionally, text classifiers adopt the *bag-of-words* model of text [30, 15, 52] and these approaches can be highly effective, despite the obviously false assumption of term independence which the methods adopt. In recent years however there have been a number of attempts to move beyond the bag-of-words model to ones which retain some dependencies between the terms [6, 44].

Comparison with *bag-of-words* Models

Table 5.8 shows the results reported in Joachims 1998 [23] compared to the results we obtained using the annotated suffix tree method. Joachims [23] experiments with four traditional classifiers and two versions of the now popular support vector machine (SVM) all based on a *bag-of-words* model. The traditional methods used are: Naive Bayes [30], Rocchio [48], k-Nearest Neighbours [14], and Decision Trees [14]. For the SVM, two different kernels are used: *polynomial* and *radial basis functions* (RBF). Joachims [23] experiments with a number of degrees of the polynomial kernel and number of different variances of the RBFs, but we consider only the best performing examples for brevity.

	Bayes	Rocchio	C4.5	k-NN	SVM (poly)	SVM (rbf)	pASTC (Wd=7)	rASTC (Wd=1)
earn	95.9	96.1	96.1	97.3	98.4	98.5	98.30	95.49
acq	91.5	92.1	85.3	92	95.2	95.3	96.74	88.46
money-fx	62.9	67.6	69.4	78.2	74.9	75.4	84.01	76.41
grain	72.5	79.5	89.1	82.2	91.3	91.9	87.89	86.08
crude	81	81.5	75.5	85.7	88.9	89	90.56	84.96
trade	50	77.4	59.2	77.4	77.3	78	81.72	71.91
interest	58	72.5	49.1	74	73.1	75	81.73	80.30
ship	78.7	83.1	80.9	79.2	86.5	86.5	82.28	84.10
wheat	60.6	79.4	85.5	76.6	85.9	85.9	72.60	59.15
corn	47.3	62.2	87.7	77.9	85.7	85.7	69.11	55.94
MacroAv	69.84	79.14	77.78	82.05	85.72	86.12	84.49	78.28
MicroAv	72	79.9	79.4	82.3	86.2	86.5	92.21	83.28

Table 5.8: Results of word-based Annotated Suffix Tree Classifier using the probabilistic significance function with depth 7 (pASTC, Wd=7) and the root significance function with depth 1 (rASTC, Wd=1) compared against the results from Joachims 1998

The results are quoted in [23] as micro-averaged precision/recall breakeven points; and we additionally calculate the macro-averaged breakeven scores for further comparison. The annotated suffix tree classifier using a probabilistic significance function (pASTC) has the best micro-averaged performance, with 92.21%, several points ahead of the the next best performer, the SVM using radial basis functions. The annotated suffix tree classifier using the root significance function (rASTC) performs worse than both versions of SVM, but performs better than all the traditional classifiers in terms of micro-averaged breakeven points. In terms of macro-averaged results, however, the ASTCs do not perform so well: the pASTC is outperformed by both versions of SVM and the rASTC is outperformed by all but the naive Bayes and C4.5.

We can understand these results by looking at the individual class scores which are arranged in descending order of class size. In the case of pASTC, the individual class performance correlates much better with the size of the class (in terms of number of training examples) than in the case of any of the other classifiers. Such individual class performance will tend to translate into a higher micro-averaged score relative to macro-averaged score. The reason pASTC performs badly on the smaller classes, relative to its performance on the larger classes is because at a depth of 7, suffix trees become naturally sparse, and with few training examples, and so the frequency counts are not large enough to give reliable probability estimates. Conversely, the quoted rASTC results are at a depth of just 1 (making it equivalent to the bag-of-words model), and as can be seen, there is less correlation between the size of a class and performance; which in turn translates into macro- and micro-scores which are much closer together.

Comparison with Word-Sequence Models

Authors in the literature mean different things by a 'word-sequences'. The more typical definition of it is followed by Peng et al. 2003 [44], in which a word-sequence is a sequence of contiguous words of some length, n; such a word-sequence is typically referred to as an $n-gram$. However, Cancedda et al. 2003 [6], define a word-sequence as a sequence of *possibly non-contiguous* terms. Hence the contiguous sequence becomes a special case of the more general word-sequence. Our ASTC models described so far all assume contiguous word- (and character-) sequences, but it would be straightforward to encompass non-contiguous sequences within our classification framework (see Section 7).

Peng et al. [44] employ what they refer to as a Chain Augmented Naive (CAN) Bayes Classifier, which extends the Bayesian framework we have already encountered (Section 4.2) to allow for the calculation of Baysian probabilities over $n-grams$ by considering them as Markov chains [14] [5] of degree n. They achieve their best performance using bi-grams ($n = 2$) for which they quote a micro-averaged F1 score of 81.52%. This greatly improves on the naive Bayes results quoted in, for example, Joachims [23], but falls well below the results achieved by our AST classifier and also the results quoted in Cancedda [6] (see below).

One of the weakness of the CAN Bayes classifier is identified by the authors themselves as lying in a uniform *prior*. But neither does our AST classifiers employ a prior (which is equivalent to a uniform prior in terms of class rankings), though the framework allows for the use of priors if, for example, we think of priors as normalisation coefficients (Section 3.1). We did experiment with such a set up, but found that the use of (non-uniform) priors was detrimental to performance and have therefore not focussed on their use in this chapter. Furthermore, Peng et al. [44] place emphasis on methods to handle null probabilities, whereas our approach is to focus only on the overlap between the query document and the class. The smoothing we perform is based on corpus frequencies and is built into the scoring mechanism as an adjustment to the importance we place on each overlap term (see Section 3.1). The result is that we do not in fact perform smoothing at all in the way that Peng et al. [44] approach it. Instead we focus entirely on *matched* terms.

	microAv	macroAv
SVM (contiguous)	91.60	84.23
SVM (non-contiguous)	91.75	84.50
pASTC (Wd=7)	92.21	84.49
rASTC (Wd=1)	83.28	78.28

Table 5.9: Micro- and Macro-Averaged Precision/Recall breakeven point comparisons between ASTC and SVM with contiguous and non-contiguous word-sequence kernels

Cancedda et al. 2003 [6] have a completely different approach to that of Peng et al. [44], but in many ways closer to ours because their method focuses entirely on the overlapping term-sequences – thereby circumventing the issue of smoothing. They apply the SVM method using a *string kernel* [53] originally developed in Lodhi et al. 2002 [33] for use in character-based classification (see Section 5.5.2); Table 5.9 shows their results compared against our two versions of the AST classifier. Cancedda et al. [6] quote precision/recall breakeven points because, they argue, this avoids confusion over the decision function thresholds, which can have a significant effect on performance. To make our work comparable to theirs, we do the same.

The word-sequences considered by Cancedda et al. [6] are only of length 2, where as our pASTC is taken at a depth of 7 (though, as we have seen, depths greater than 2 return only modest improvements). Cancedda et al. [6] found that their performance diminished when using word-sequences of length greater than 2, but as we have seen in Section 5.4.3, the actual variation in performance with depth (or word-sequence length) can also depend on factors such as the weighting mechanism, and indeed, the decision function or its threshold. Cancedda et al. [6] use either a binary weighting or an IDF (inverse document frequency) [3] weighting scheme, while we use a more sophisticated method which gives a better reflection of the significance of a particular word sequence match between the query document and the class; and it may be this that accounts for a slightly better performance from our method. Nevertheless, the results reported in Cancedda et al. [6] are very good; there are strengths to their approach which are so far not modelled in ours. For example, despite their more basic weighting mechanism, their method employs a more general matching scheme in that it considers non-contiguous words. Such an approach is able to ignore possibly noisy terms and identify only those that are key to the classification of a document. Furthermore, SVMs perform a form of internal feature selection (as part of the training process, they retain only the features which play the more important role in separating classes) and this too may help eliminate noisy terms which may otherwise confuse classification. Indeed, a promising line of development would be an integration of the relative strengths of their methods and the methods described in this thesis (see Chapter 7).

5.5.2 Character-Based Classifiers

Character-based text classifiers are not as common in the literature as word-based approaches, and are even less commonly applied to the Reuters corpus because it is generally considered to be more amenable to word-based analysis.

However, in recent years, there are growing efforts at applying character-based models to text classification and indeed to Reuters. Unfortunately, methods based on the bag-of-words model cannot be effectively applied to character-based analysis, but methods based on word-sequences can.

Hence Peng et al. 2003 [44] apply the same method, which we discussed in the previous section, to character-sequences, again with emphasis on smoothing techniques. For their best performing method they quote a micro-averaged F1 measure of 80.32%, which is again significantly outperformed by our ASTC, with either a root significance function (at 92.25%) or a probabilistic significance function (at 91.77%).

Similar results are achieved by Lodhi et al. 2002 [33], the originators of the approach later applied to word-sequences by Cancedda et al. [6] (Section 5.5.1), whom we discussed in the previous section. Lodhi et al. [33] apply the SVM framework to a kernel defined over possibly non-contiguous character sequences in the same way as in Cancedda et al. [6]. So, like us, they consider overlapping sequences of characters, but unlike us, use a constant weight for each overlapping feature. This results in a best-case macro-averaged F1 score of 77.3% over the largest ten Reuters classes while using (possibly non-contiguous) character sequences of length 4. This results is well below our best macro-averaged F1 scores of 84.33% for a probabilistic significance function (Table 5.7) and 84.86% for a root significance function (Table 5.5). Interestingly, Lodhi et al. [33] also implemented their own word kernel SVM for comparison purposes and found that it significantly outperformed their own character sequence kernel. It may be explanatorily important to note that they use a TF-IDF weighting scheme on their word kernel and consequently achieve a macro-averaged F1 of 85.29%, which is slightly better than any macro-averaged F1 score we achieved; unfortunately they do not quote a micro-averaged score, and therefore we cannot fully compare our work to theirs.

Finally, we would like to draw attention to Slonim et al. 2003 [54], whose work can be seen as parallel to ours because it is also based on the suffix tree data structure. They use smoothing techniques based on work by Bejerano and Gill 2001 [4] to derive what they call a Prediction Suffix Tree (PST) which associates a probability with every possible character sequence. This PST is used to evaluate the probability of each character in a document given a class; and using Bayes' rule they calculate from this the probability of a class. The focus of Slonim et al. [54] is on pruning the tree so that the shortest possible preceding sequence is used to evaluate the probability of a character, resulting in a suffix tree model which holds only a minimal set of relevant suffixes. The process they use is equivalent to feature selection using mutual information, but using probabilities conditioned not only on classes but also on the sequences preceding a particular character feature. We also experimented with entropy based feature selection methods in similar ways, but with an aim to selecting full (maximum possible) length branches in the profile tree which were most indicative of classes, hoping to prune away noise in the model, but found most of our attempts tended to reduce performance. This is not to say that feature selection is not possible, but only that we did not achieve a successful approach. The important point is that our position is opposite to that of Slonim et al. because we seek the longest possible matches between documents and classes rather than seeking the shortest possible predictive sequence.

Slonim et al. experiment on the largest ten classes from Reuters, as we do, and quote a micro-averaged precision and recall of 95% and 87% respectively, from which they impute a likely break-even point of 87% or above. These precision and recall values certainly equate to micro-averaged F1 score of 90.82%, which is slightly lower than our results.

5.6 Conclusion

This chapter has extended the annotated suffix tree method to general text classification and has shown that its performance competes well with state-of-the-art techniques and outperforms most other methods reported in the literature.

We have shown that the method presented in this thesis is agnostic as to whether it is applied at the level of characters or words; what matters more is the domain under consideration, the consequent properties of the data, and the appropriate choice of significance function. Hence we have seen that for data sets such as Reuters, which is highly amenable to word-based analysis, the word-based ASTC performs better than the character-based ASTC - exactly as one would expect. But in other domains, such as spam email, character-based analysis is more appropriate, and one of the great strengths of the ASTC method is that it can present a uniform solution in both cases.

That said, it is also interesting that with data sets such as Reuters, the word-based ASTC only slightly outperforms its character-based equivalent. And the character-based ASTC outperforms many alternative word-based classifiers. Hence the character-based ASTC is arguably preferable in most situation because the slight loss in performance is significantly outweighed by the benefits of a uniform approach, which would reduce the need for domain- or language-specific preprocessing of the data, such as the compilation of stop-word lists and the employment of stemming or lemmatisation algorithms.

We have also seen that in recent years other researchers have begun to see the advantages that character-level treatment of text may afford. That such an approach can be highly effective and competes successfully with the more traditional word-based approaches is reflected not only by the successes of our method but also in the relative success of all the methods we have reviewed in this chapter.

Regarding alternative approaches, there is certainly overlap between our approach and those of Lodhi al [33], Cancedda et al [6] and Slonim et al. [54]. The former two evaluate a set of sequences which overlap between the document and the class, just as we do; and the latter, like us, use a variant of the suffix tree data structure – evaluating the smoothed probability of all characters in a query document.

One of the key differences between our approach is the overlap mass score (OMS) method which, along with the appropriate significance function is able to take into account aspects of the significance of a term which are ignored by the approaches of the others. Central to this is, of course, our interpretation of the suffix tree because it allows us to efficiently evaluate overlaps and conditional probabilities over all sequences and all classes and hence allows the construction of these significance functions in the straight forward ways that we demonstrated in Chapters 2 and 3.

The clustering method for selecting class membership is also important to the success of our approach for the reasons already given in Section 5.2.1. However, a weakness of the clustering decision method is that it always selects at least one class, even if none of the classes are in fact the true class. This weakness may certainly be addressed, though we have not done so here. For example, one may use a set of negative examples which belong to none of the considered classes (such a ‘negative class’ in fact exists in the Reuters data set); if this negative class is the only member of the positive cluster, then we could conclude that the document does not belong to any of the classes. An alternative approach is to utilise a combination of the class-specific threshold and the clustering decision function so that a class must be a member of the positive cluster *and* score above a certain threshold. However, in many cases, such issues will not be a concern, and there will often be the assumption that a document must belong to at least one the training classes.

Regarding the clustering decision method itself, an important and useful result is that we found that there seems to be a maximal cluster bias value above which marginal improvement in performance was close to zero or negative. This value, of approximately 4.0, was applicable for all the flavours of the ASTC we experimented with and may provide a good rule-of-thumb when using an ASTC in a real-world context. It would mean that such a bias could be used as a default value without needing extensive investigation of the particular data and domain.

As we expected, we found that we needed less depth in the tree profiles in order to obtain a maximal performance. Indeed, whereas we did not, in the spam domain, reach a maximal performance depth, we have seen in this chapter that our conjecture (in Section 4.5.2) that such a depth would be reached was true. We saw in Figure 5.9 that performance, for the character-based ASTC using the root significance function, peeked at a depth of 5 and then began to fall; while the performance of the probabilistic significance function, though it did not fall, began to stabilise at a maximal performance level at around a depth of 5.

The most surprising and inexplicable result was the poor performance of the root significance function when applied to word-based ASTCs. Not only was performance well below its own character-based equivalent but also against the performance of both types of probabilistic ASTCs. Unfortunately, we are unable to adequately explain this result at present.

Chapter 6

Classification Feature Evaluation and Visualisation

In this chapter we present examples of the application of the suffix tree text model to document highlighting using the approach described in detail in Chapter 3 Section 3.3.

The effectiveness of document highlighting cannot be assessed in the same quantitative way that class prediction is assessed. User studies may be conducted to determine the usefulness of certain approaches, but we do not pursue such a study here. Instead we present examples of results and point to certain advantages that are afforded by using the ASTC model for such tasks.

The main motivation for document highlighting is to provide a user with some justification for a particular classification decision by providing a visualisation of the importance of each term in the document relative to a particular class, which we will refer to as the *relative class*.

Such a facility has particular benefit in certain domains. For example, the sponsors of the work presented in this thesis envisioned its use in expertise location, where each expert is represented as a class. The process of seeking an expert begins with a document (or simply a typed query) which contains material that an expert seeker requires help in understanding. The method first 'classifies' this document (or query) into one or more classes (i.e. experts) and the seeker is thereby presented with a short-list of possible candidate experts. The role of document highlighting would then be to allow the seeker to compare the document to each short-listed expert (i.e. class) in order to obtain a visualisation of the reasons for the inclusion of that expert in the short-list. If the highlighted terms are those that the seeker is in fact interested in, there is then good justification to contact that expert.

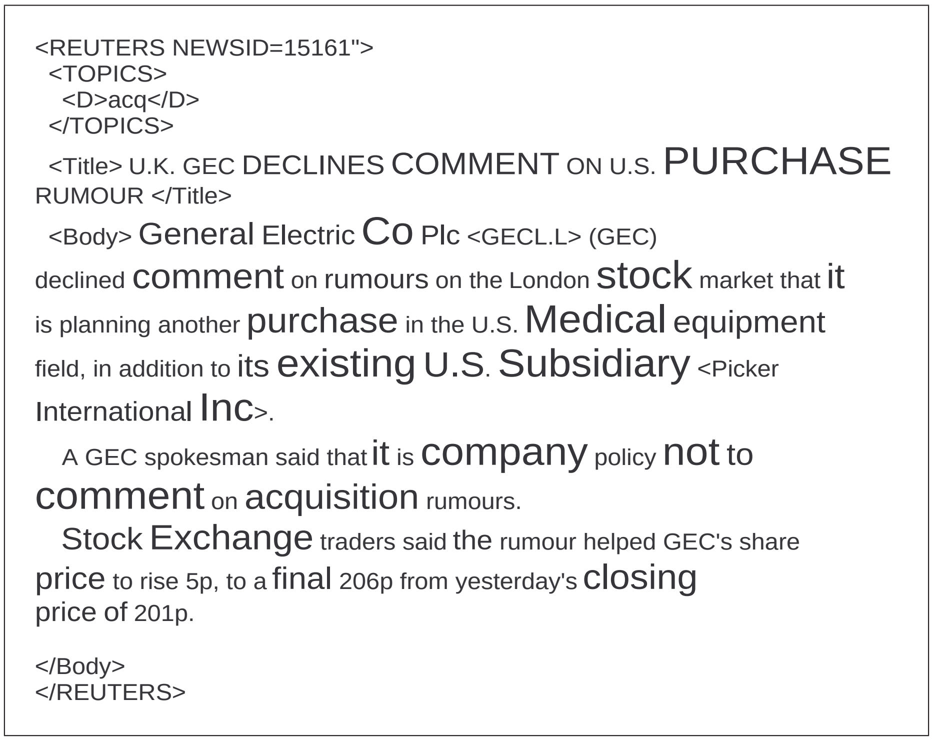
In a broader context, the facility may be used to gain insight into the most significant terms in a particular class and the terms in the document which play a significant role in any classification decision. To some degree we can also use the technique to visualise the classification process because we are able to see the way in which different parameter settings of the classifier effect its performance. However, this latter function is limited because it is difficult to perform a comprehensive comparison of all documents against all classes. Rather, the method can simply give us some clues about certain significant terms which may be used to guide further investigation.

Thus, our concern in this chapter is not to investigate all the effects of parameter settings or even to experiment with all possible highlighting policies, both of which are too numerous, but to use the method to point to specific characteristics of the classifier which are evident under certain parameter settings.

All the examples in this chapter use the same highlighting policy: the terms in the document are each evaluated relative to a particular class and then split by score into 10 partitions, with the top 4 partitions 'highlighted' by ascending order of text size. Hence the terms which are in the top partition (effectively the top 10% of term scores for this document when compared against the relative class) have the largest font size and those below the 60^{th} percentile are left in the original font size.

For character-based classifiers, we additionally perform *smoothing* within word boundaries to assist visual assessment of the results of highlighting. This is done by simply assigning to each word the sum of the scores received by each of its characters.

The results bear some resemblance to "Tag Clouds" or "Word Clouds", which emerged outside the academic literature and became popularised with the advent of social networking sites; and have more recently attracted some academic interest [22]. Tag Clouds are certainly similar in that they extract from a body of text – be it a single web page or an entire web site – a set of the most frequently occurring terms and present them as indicative of the content of the text and highlight them by size or colour according to their frequency. Some more sophisticated techniques may use methods other than simple frequency counts, but all of them are effectively aiming to evaluate the body of text as distinct (from any other body of text). By contrast, the motivation of our document-to-class highlighting method is



```
<REUTERS NEWSID=15161">
  <TOPICS>
    <D>acq</D>
  </TOPICS>
  <Title> U.K. GEC DECLINES COMMENT ON U.S. PURCHASE
RUMOUR </Title>
  <Body> General Electric Co Plc <GECL.L> (GEC)
declined comment on rumours on the London stock market that it
is planning another purchase in the U.S. Medical equipment
field, in addition to its existing U.S. Subsidiary <Picker
International Inc>.
    A GEC spokesman said that it is company policy not to
comment on acquisition rumours.
    Stock Exchange traders said the rumour helped GEC's share
price to rise 5p, to a final 206p from yesterday's closing
price of 201p.

</Body>
</REUTERS>
```

Figure 6.1: Reuters document belonging to ‘acq’ class highlighted against the ‘acq’ class using a word-based probabilistic ASTC of depth 4 using the probabilistic scoring function (pASTC-W4).

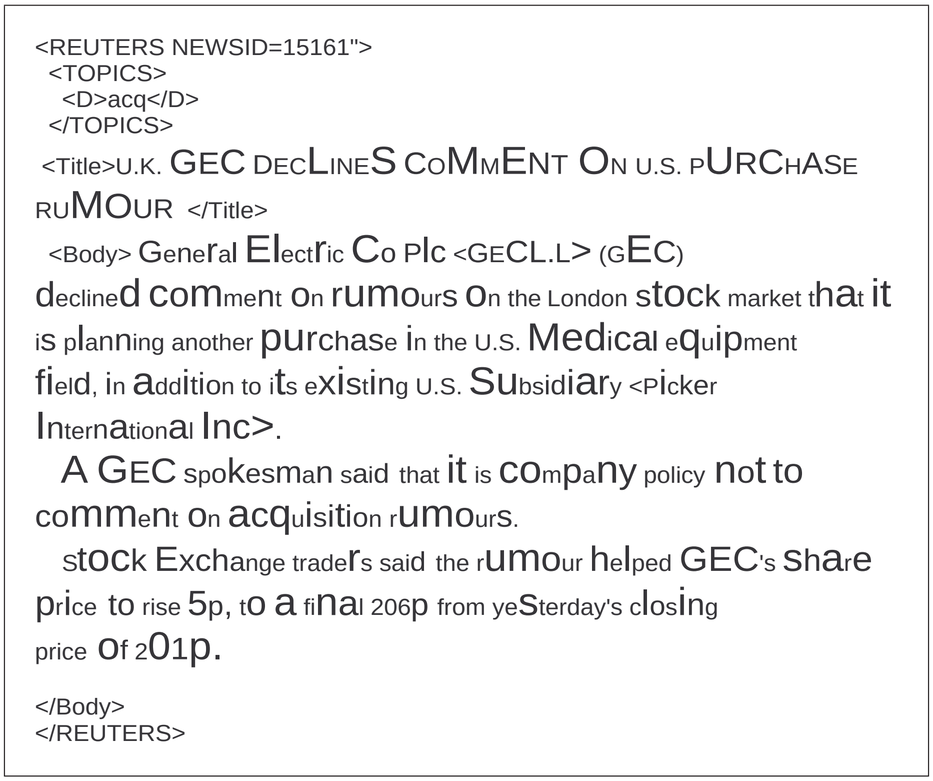
completely different in that it aims to provide an evaluation of a body of text as it is compared *against a particular class*.

The next three sections look at the highlighting that results from using character-based and word-based classifiers of different depths and against different relative classes.

6.1 Effect of Varying the Relative Class

Our first example, presented in Figure 6.1, shows the same news story as that shown in Figure 5.2, taken from the ‘acq’ class of the ModApte test set. The story is highlighted against the ‘acq’ class using a word-based probabilistic AST classifier of depth 4 (pASTC-W4).

The first point to note is that the importance of a term is a function not only of it, but also of its context. For example, though the term ‘rumours’ occurs in the second line of the first paragraph of the body of the text and in the second line of the second paragraph of the body, it is highlighted only in the



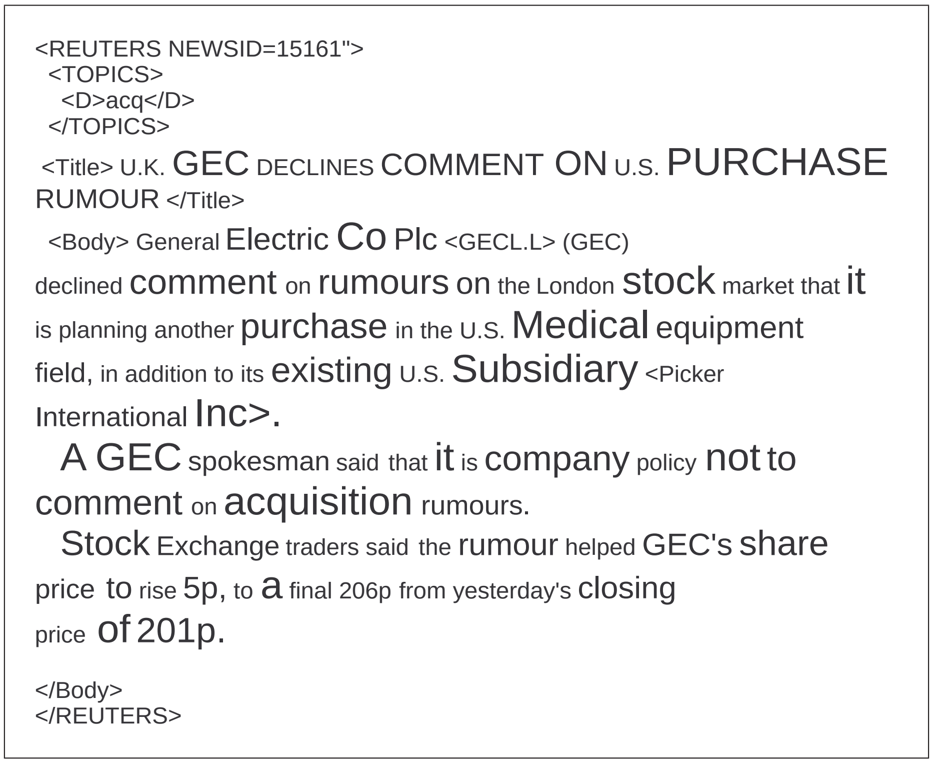
```
<REUTERS NEWSID=15161">
  <TOPICS>
    <D>acq</D>
  </TOPICS>
 <Title>U.K. GEC DECLINES COMMENT ON U.S. PURCHASE
RUMOUR </Title>
  <Body> General Electric Co Plc <GECL.L> (GEC)
declined comment on rumours on the London stock market that it
is planning another purchase in the U.S. Medical equipment
field, in addition to its existing U.S. Subsidiary <Picker
International Inc>.
    A GEC spokesman said that it is company policy not to
comment on acquisition rumours.
    Stock Exchange traders said the rumour helped GEC's share
price to rise 5p, to a final 206p from yesterday's closing
price of 201p.

</Body>
</REUTERS>
```

Figure 6.2: Reuters document belonging to ‘acq’ class highlighted against the ‘acq’ class using a character-based ASTC of depth 8 using the probabilistic scoring function (pASTC-C8); characters are split into 10 partitions with the top 4 highlighted in four ascending text sizes.

first case. Such context sensitivity would be impossible for a traditional bag-of-words model of text.

Secondly, the significance of certain terms such as ‘purchase’ and ‘acquisition’ is not surprising considering that all the stories in this class are about one company acquiring another, but terms such as ‘medical’ and ‘subsidiary’ are more surprising. A quick investigation motivated by this finding shows that in fact both these latter two terms are quite highly indicative of the ‘acq’ class: ‘subsidiary’ occurs 425 times in the training data of which 63% are in documents belonging to the ‘acq’ class; while ‘medical’ occurs 93 times, of which 60% are in ‘acq’.

Figure 6.2 shows the highlighting of the same document using a character-based probabilistic classifier of depth 8 (pASTC-C8). To make it easier for us to see what is happening and compare it against the word-based highlighting, Figure 6.3 shows the same highlighting, but now *smoothed* so that each word is given the sum of the scores of its individual characters.



<REUTERS NEWSID=15161">
<TOPICS>
<D>acq</D>
</TOPICS>
<Title> U.K. GEC DECLINES COMMENT ON U.S. PURCHASE RUMOUR </Title>
<Body> General Electric Co Plc <GECL.L> (GEC) declined comment on rumours on the London stock market that it is planning another purchase in the U.S. Medical equipment field, in addition to its existing U.S. Subsidiary <Picker International Inc>.

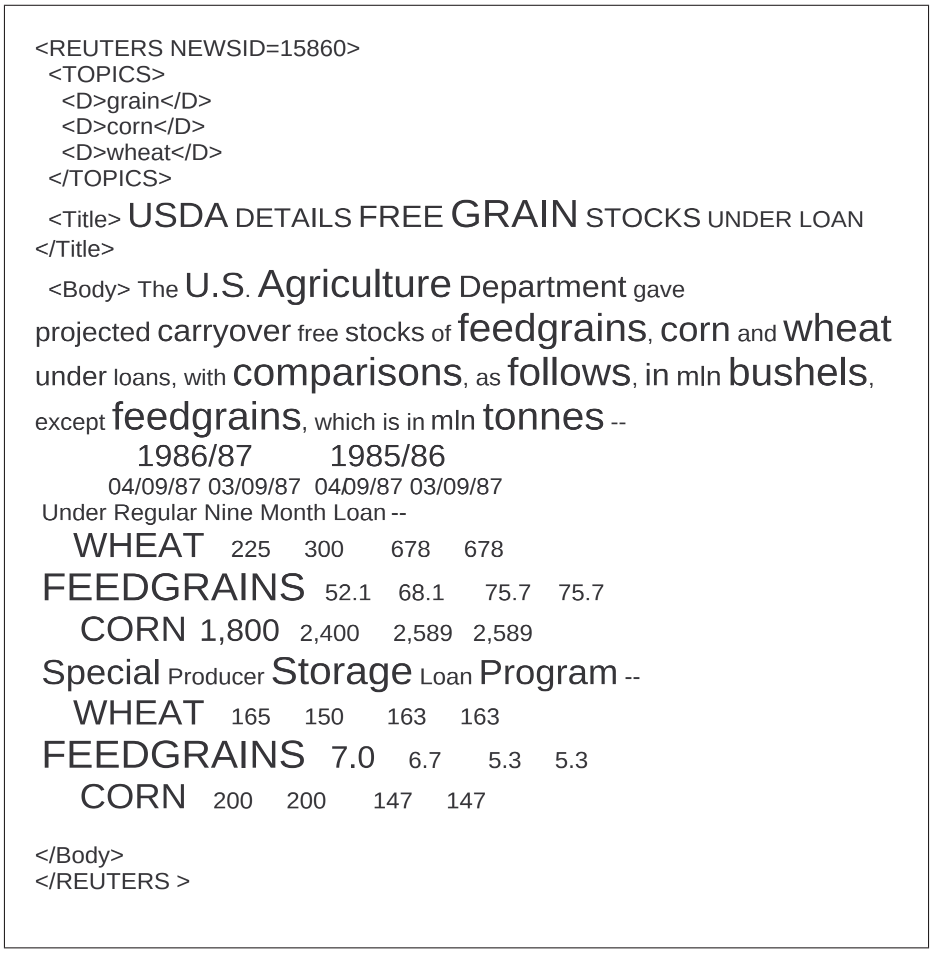
A GEC spokesman said that it is company policy not to comment on acquisition rumours.

Stock Exchange traders said the rumour helped GEC's share price to rise 5p, to a final 206p from yesterday's closing price of 201p.
</Body>
</REUTERS>

Figure 6.3: Reuters document belonging to 'acq' class highlighted against the 'acq' class using a character-based ASTC of depth 8 using the probabilistic scoring function (pASTC-C8). Characters are scored and then the scores are smoothed within word boundaries such that each word is assigned the sum of the score for its individual characters. The resulting word scores are then split into 10 partitions with the top 4 highlighted in four ascending text sizes.

We can now see more easily the similarities and difference in the highlighting that results from the use of character-based and word-based classifiers. Many of the important terms turn out to be exactly the same - such as 'purchase', 'medical', 'subsidiary' and 'acquisition'. But the term 'rumour' is now highly significant whereas before it was much less so. The reason for this demonstrates one of the strengths of the character-based approach. The word-based approach fails to pick up the importance of this term to the 'acq' class because it occurs only 16 times in the training data out of which only 4 are in the 'acq' class, whereas the term 'rumor' (the American spelling) occurs 102 times in the training data, of which well over 50% are in the 'acq' class.

With this in mind, if we look back to Figure 6.2, we can see that for the occurrence of the term 'rumour' it is mainly the character sequence, 'umo', in the middle of the term which lends weight to the

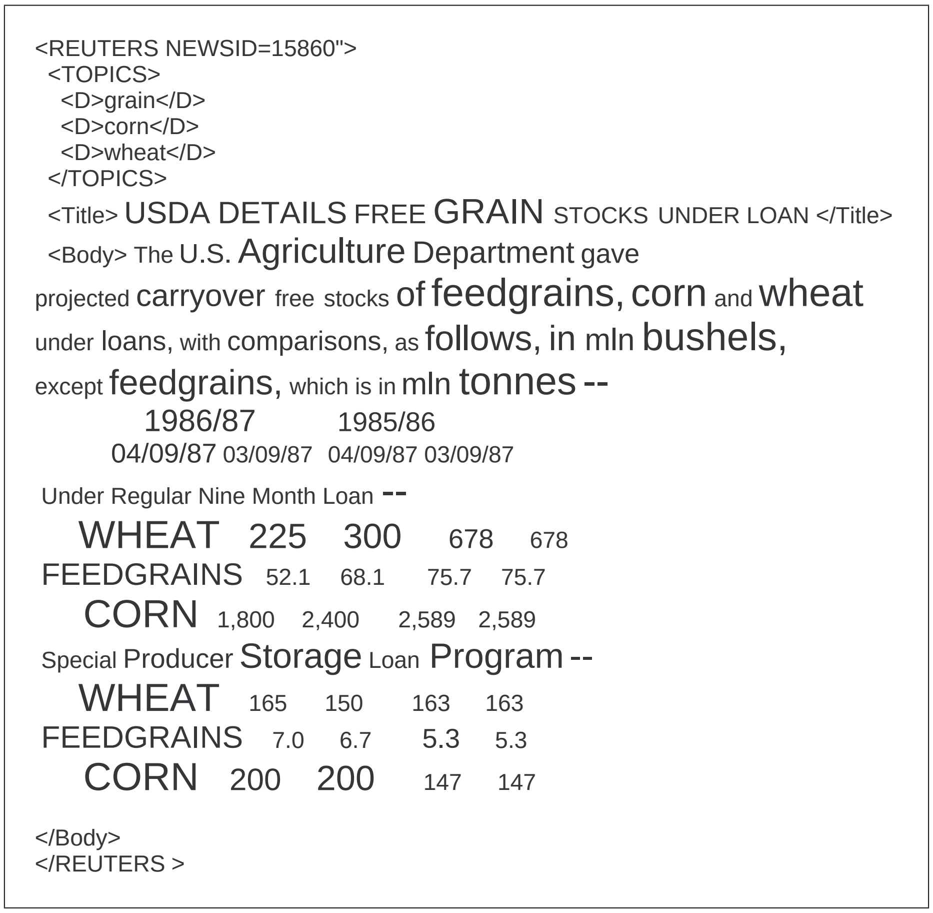


```
<REUTERS NEWSID=15860>
  <TOPICS>
    <D>grain</D>
    <D>corn</D>
    <D>wheat</D>
  </TOPICS>
  <Title> USDA DETAILS FREE GRAIN STOCKS UNDER LOAN
</Title>
  <Body> The U.S. Agriculture Department gave
projected carryover free stocks of feedgrains, corn and wheat
under loans, with comparisons, as follows, in mln bushels,
except feedgrains, which is in mln tonnes --
               1986/87          1985/86
           04/09/87 03/09/87  04/09/87 03/09/87
 Under Regular Nine Month Loan --
     WHEAT    225    300     678    678
 FEEDGRAINS   52.1   68.1    75.7   75.7
      CORN 1,800  2,400    2,589  2,589
 Special Producer Storage Loan Program --
     WHEAT    165    150     163    163
 FEEDGRAINS   7.0    6.7     5.3    5.3
      CORN    200    200     147    147

</Body>
</REUTERS >
```

Figure 6.4: Reuters document belonging to the ‘grain’, ‘corn’ and ‘wheat’ classes highlighted against the ‘grain’ class using a word-based ASTC of depth 4 using the probabilistic scoring function (pASTC-W4); terms are split into 10 partitions with the top 4 highlighted in four ascending text sizes.

term’s overall score.

Of course, certain pre-processing techniques commonly used with word-based classifiers, such as stemming, would be able to handle such variations in text, but others, such as spelling mistakes are not so readily addressed. Moreover, a character-based approach cannot only function well without the need for pre-processing such as stemming and punctuation removal, but in many cases is able to benefit from such indicators as formatting and punctuation, which might be removed by pre-processing - a point we have previously argued, but are now able to visualise. For example, compare the highlighting

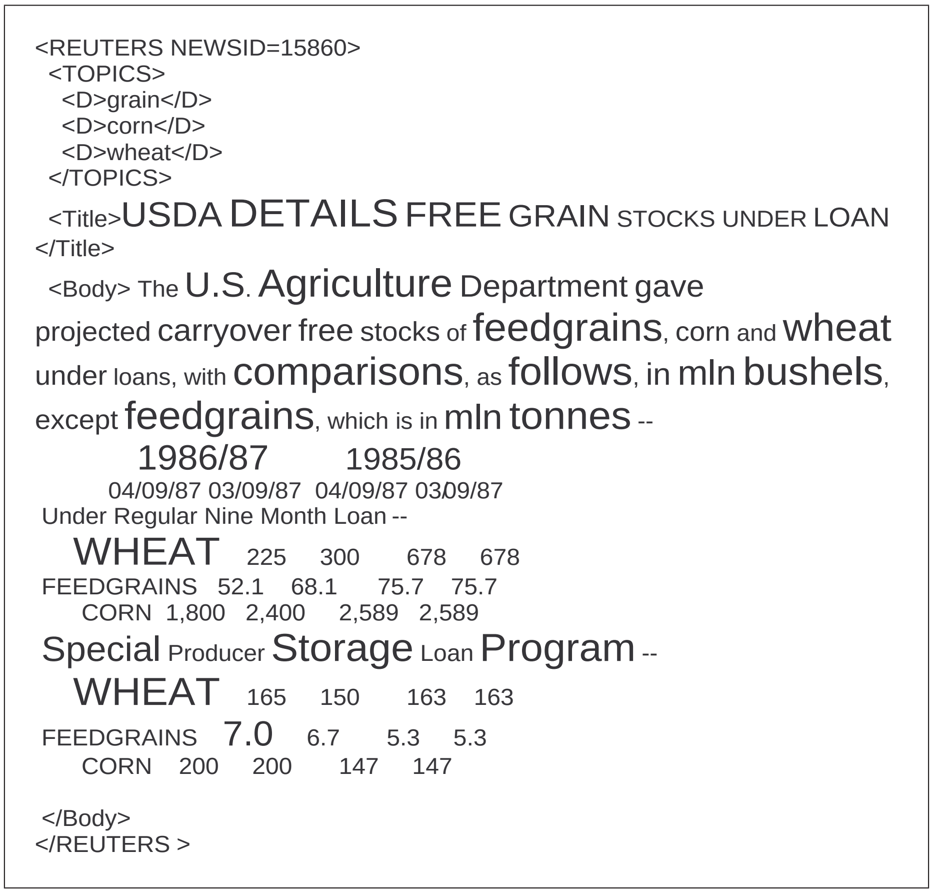


```
<REUTERS NEWSID=15860">
  <TOPICS>
    <D>grain</D>
    <D>corn</D>
    <D>wheat</D>
  </TOPICS>
  <Title> USDA DETAILS FREE GRAIN STOCKS UNDER LOAN </Title>
  <Body> The U.S. Agriculture Department gave
projected carryover free stocks of feedgrains, corn and wheat
under loans, with comparisons, as follows, in mln bushels,
except feedgrains, which is in mln tonnes --
              1986/87           1985/86
          04/09/87 03/09/87  04/09/87 03/09/87
 Under Regular Nine Month Loan --
    WHEAT   225    300     678    678
 FEEDGRAINS   52.1    68.1     75.7    75.7
     CORN  1,800    2,400     2,589   2,589
 Special Producer Storage Loan Program --
    WHEAT    165     150      163     163
 FEEDGRAINS    7.0     6.7      5.3     5.3
     CORN   200    200      147     147

</Body>
</REUTERS >
```

Figure 6.5: Reuters document belonging to the ‘grain’, ‘corn’ and ‘wheat’ classes highlighted against the ‘grain’ class using a character-based ASTC of depth 8 using the probabilistic scoring function (pASTC-C8). Character scores are smoothed within word boundaries and are then split into 10 partitions with the top 4 highlighted in four ascending text sizes.

that results from the word- and character-based classifiers as shown in Figures 6.4 and 6.5 (again the character-based highlighting is smoothed for easier comparison). Both examples show a news story we have seen before in Figure 5.1 belonging to the ‘corn’, ‘grain’ and ‘wheat’ classes. The document is now highlighted against the ‘grain’ class.

We can see that both classifiers are identifying very similar terms as significant and indeed, all the ones which we would expect them to identify - such as ‘grain’, ‘agriculture’ etc.. But additionally, the character-based approach is able to identify formatting marks such as the double dash (‘– –’) which is commonly used to arrange the tabulated information frequently appearing in the ‘corn’, ‘grain’ and

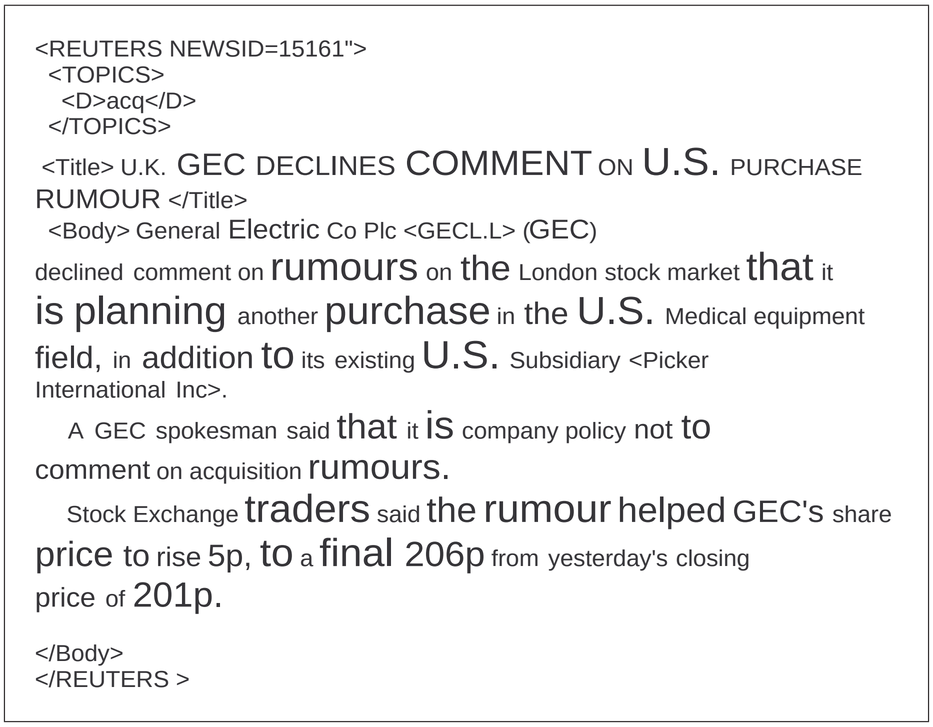


```
<REUTERS NEWSID=15860>
  <TOPICS>
    <D>grain</D>
    <D>corn</D>
    <D>wheat</D>
  </TOPICS>
  <Title>USDA DETAILS FREE GRAIN STOCKS UNDER LOAN
</Title>
  <Body> The U.S. Agriculture Department gave
projected carryover free stocks of feedgrains, corn and wheat
under loans, with comparisons, as follows, in mln bushels,
except feedgrains, which is in mln tonnes --
              1986/87          1985/86
          04/09/87 03/09/87  04/09/87 03/09/87
 Under Regular Nine Month Loan --
    WHEAT    225    300      678    678
 FEEDGRAINS   52.1    68.1      75.7    75.7
      CORN  1,800   2,400     2,589   2,589
 Special Producer Storage Loan Program --
    WHEAT    165    150      163    163
 FEEDGRAINS   7.0    6.7      5.3     5.3
      CORN    200    200      147     147

 </Body>
</REUTERS >
```

Figure 6.6: Reuters document belonging to the 'grain', 'corn' and 'wheat' classes highlighted against the 'wheat' class using a word-based ASTC of depth 4 using the probabilistic scoring function (pASTC-W4); terms are split into 10 partitions with the top 4 highlighted in four ascending text sizes.

'wheat' classes; also, we see that numerals are more readily used as class indicators by the character-based classifier.

The other main advantage of document highlighting is that it allows us to see term significance relative to a particular class. The news story shown in Figure 5.1 belongs to three classes and we have so far only seen it highlighted against one of these: 'grain'. Figure 6.6 shows the story highlighted against the 'wheat' class, where we can now see that terms such as 'wheat' become far more significant. Hence, a user of the system who is more interested specifically in 'wheat' rather than in the more generic 'grain' would be able to judge that the 'wheat' class is more relevant to their needs (obviously, in some



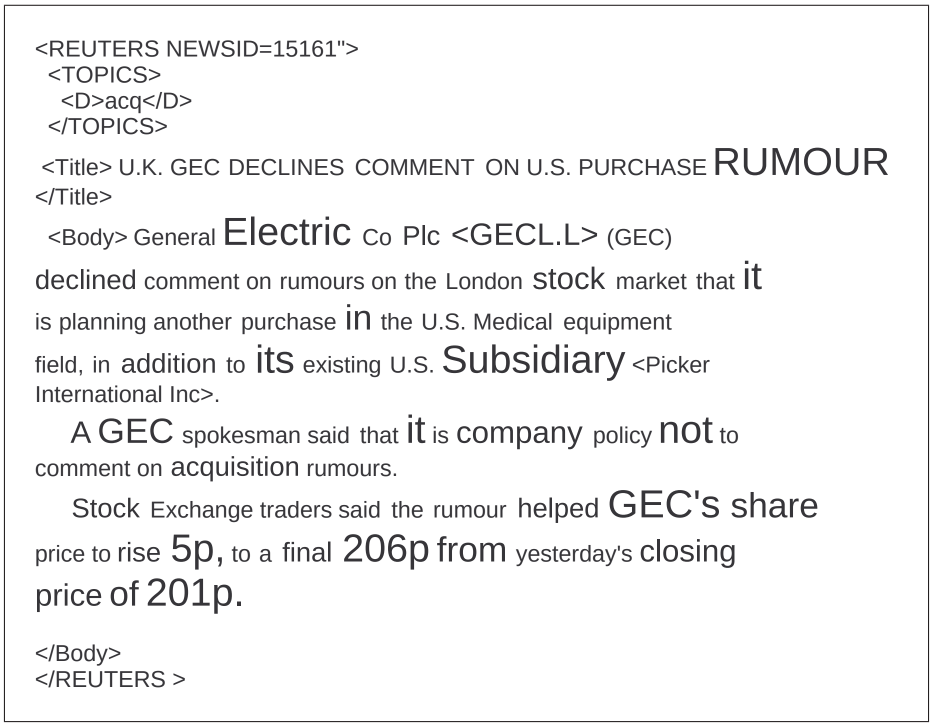
```
<REUTERS NEWSID=15161">
  <TOPICS>
    <D>acq</D>
  </TOPICS>
 <Title> U.K. GEC DECLINES COMMENT ON U.S. PURCHASE
RUMOUR </Title>
  <Body> General Electric Co Plc <GECL.L> (GEC)
declined comment on rumours on the London stock market that it
is planning another purchase in the U.S. Medical equipment
field, in addition to its existing U.S. Subsidiary <Picker
International Inc>.
    A GEC spokesman said that it is company policy not to
comment on acquisition rumours.
    Stock Exchange traders said the rumour helped GEC's share
price to rise 5p, to a final 206p from yesterday's closing
price of 201p.

</Body>
</REUTERS >
```

Figure 6.7: Reuters document belonging to ‘acq’ class highlighted against the ‘corn’ class using a character-based ASTC of depth 8 using the probabilistic scoring function (pASTC-C8). Characters are scored and then scores are smoothed within word boundaries such that each word is assigned the sum of the score for its individual characters. The resulting word scores are then split into 10 partitions with the top 4 highlighted in four ascending text sizes.

domains, such as that of expertise location, the name of the class (i.e. the expert’s name) may not always so obviously indicate the subject most relevant to it!).

The other possible case is when the relative class is one that the document does not belong to. Figure 6.7 shows our running example from the acquisition class highlighted against the ‘corn’ class. Against this class, terms such as ‘rumours’ and ‘comment’ are not highly significant in any context; and overall, there is less in the document that is relevant to the ‘corn’ class. However, the terms, “purchase”, “stock market” and “Stock Exchange traders” are relevant to both classes, though less so in the case of ‘corn’; and although “price” is equally relevant to both classes, “closing price”, a concept that is special to share markets, is much more significant to the ‘acq’ class. All this generally concurs with our knowledge of these two classes.

Comparing the document against a class such as ‘earn’, which also tends to focus on the stock

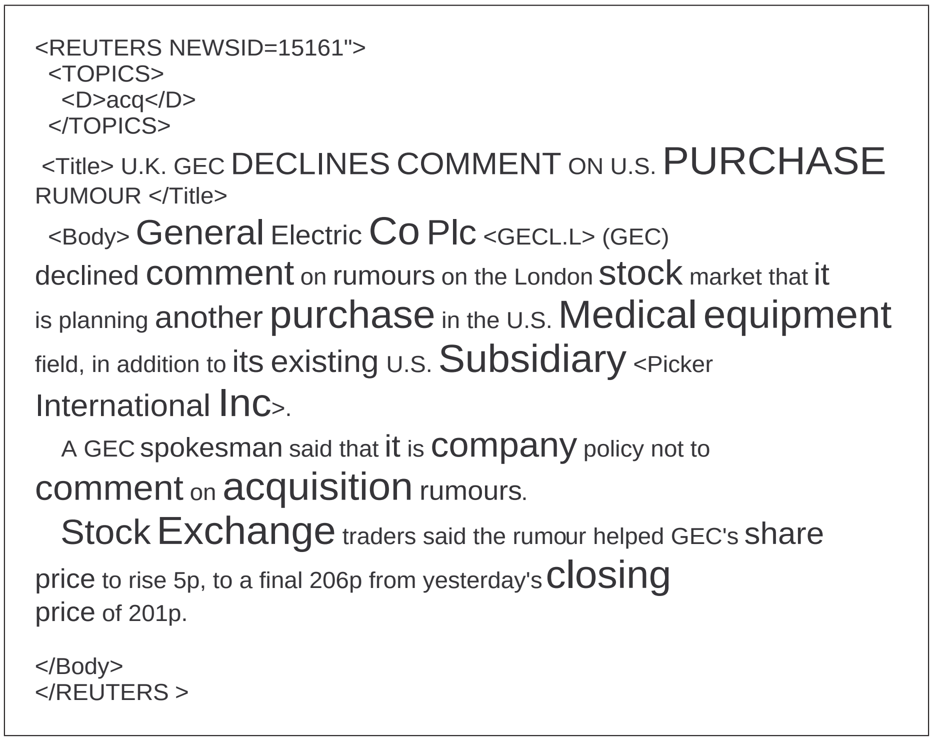


<REUTERS NEWSID=15161">
<TOPICS>
<D>acq</D>
</TOPICS>
<Title> U.K. GEC DECLINES COMMENT ON U.S. PURCHASE RUMOUR
</Title>
<Body> General Electric Co Plc <GECL.L> (GEC)
declined comment on rumours on the London stock market that it
is planning another purchase in the U.S. Medical equipment
field, in addition to its existing U.S. Subsidiary <Picker
International Inc>.
A GEC spokesman said that it is company policy not to
comment on acquisition rumours.
Stock Exchange traders said the rumour helped GEC's share
price to rise 5p, to a final 206p from yesterday's closing
price of 201p.
</Body>
</REUTERS >

Figure 6.8: Reuters document belonging to ‘acq’ class highlighted against the ‘earn’ class using a character-based ASTC of depth 8 using the probabilistic scoring function (pASTC-C8). Characters are scored and then scores are smoothed within word boundaries such that each word is assigned the sum of the score for its individual characters. The resulting word scores are then split into 10 partitions with the top 4 highlighted in four ascending text sizes.

market, yields the different result shown in Figure 6.8.

Here, terms such as “stock market” and “closing price” are just as important as they are for the ‘acq’ class, but terms such “rumours” and “purchase” are much less significant. Such a result correlates well with our understanding of the classes: in the reporting of acquisitions, rumours may be important, but they would be irrelevant to the reporting of a company’s announcement of its earnings, which would be made publicly; hence, it would be unlikely that a company would have a policy of not commenting on its earnings.



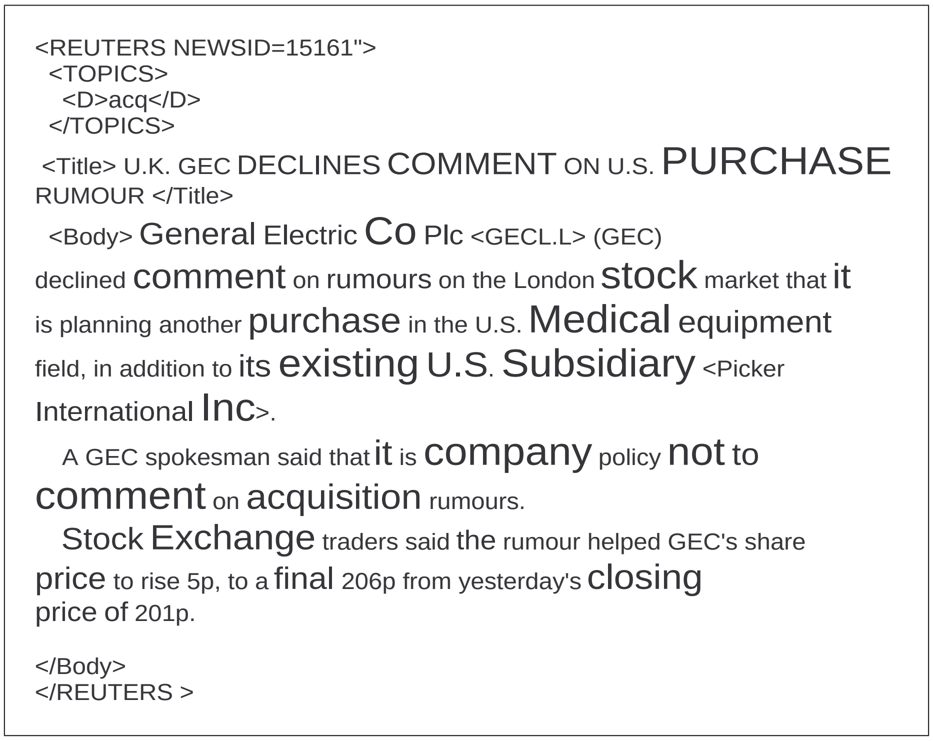
<REUTERS NEWSID=15161">
  <TOPICS>
    <D>acq</D>
  </TOPICS>
 <Title> U.K. GEC DECLINES COMMENT ON U.S. PURCHASE RUMOUR </Title>
  <Body> General Electric Co Plc <GECL.L> (GEC) declined comment on rumours on the London stock market that it is planning another purchase in the U.S. Medical equipment field, in addition to its existing U.S. Subsidiary <Picker International Inc>.
    A GEC spokesman said that it is company policy not to comment on acquisition rumours.
    Stock Exchange traders said the rumour helped GEC's share price to rise 5p, to a final 206p from yesterday's closing price of 201p.
</Body>
</REUTERS >

Figure 6.9: Reuters document belonging to ‘acq’ class highlighted against the ‘acq’ class using a word-based probabilistic ASTC of depth 1 using the probabilistic scoring function (pASTC-W1); terms are split into 10 partitions with the top 4 highlighted in four ascending text sizes.

6.2 Effect of Depth Variation

We can also use highlighting to observe the effect of changes in the depth - for both word-based and character-based classifiers. So far all stories have been highlighted using word-based classifiers of depth 4 and character based classifiers of depth 8.

With word-based classifiers, the impact of depth variation is small. Figure 6.9 shows our running example story from the ‘acq’ class highlighted using a word-based classifier of depth 1. We can see now that the effects of context are completely removed so that the term ‘rumours’ gains the same weighting regardless of where it appears. The same may be seen for other repeated terms such as ‘stock’, ‘comment’ and ‘price’. Of course, this does not say anything about whether such a change in weighting results in better classification – that would be better determined by systematic testing over a large numbers of stories – but it does show the change in the relative importance attached to certain terms as they



<REUTERS NEWSID=15161">
<TOPICS>
<D>acq</D>
</TOPICS>
<Title> U.K. GEC DECLINES COMMENT ON U.S. PURCHASE RUMOUR </Title>
<Body> General Electric Co Plc <GECL.L> (GEC) declined comment on rumours on the London stock market that it is planning another purchase in the U.S. Medical equipment field, in addition to its existing U.S. Subsidiary <Picker International Inc>.

A GEC spokesman said that it is company policy not to comment on acquisition rumours.

Stock Exchange traders said the rumour helped GEC's share price to rise 5p, to a final 206p from yesterday's closing price of 201p.

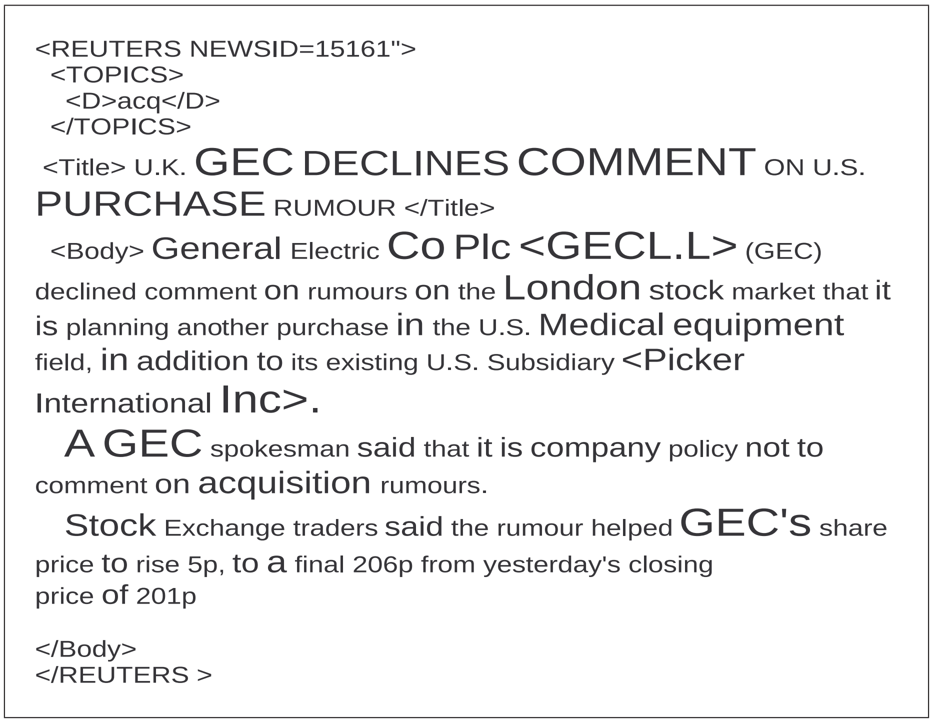
</Body>
</REUTERS >

Figure 6.10: Reuters document belonging to ‘acq’ class highlighted against the ‘acq’ class using a word-based probabilistic ASTC of depth 8 using the probabilistic scoring function (pASTC-W8); terms are split into 10 partitions with the top 4 highlighted in four ascending text sizes.

appear in the text.

Inspecting many such examples shows that, generally, depths greater than 4 have very little impact on the relative importance that the classifier places on contextual terms within the document. To demonstrate this with our running example, we present in Figure 6.10 the story highlighted using a classifier of depth 8. As can be seen, there is no difference in the highlighting in this case and in the case of a classifier of depth 4. Of course, this (lack of difference) is only so given our current highlighting policy (of dividing scores into 10 partitions and highlighting the top four in increasing order of text size); any change in depth is quite likely to make at least some difference, however small, to the final scores achieved by each term in the document, but such a change may be too small (as it is in this case) to make a difference to the highlighting or indeed to any classification decision.

Character-based classifiers are more sensitive to depth changes, particularly at low depths. Indeed at a depth of 1, we would expect that the classifier will have little chance of identifying any features



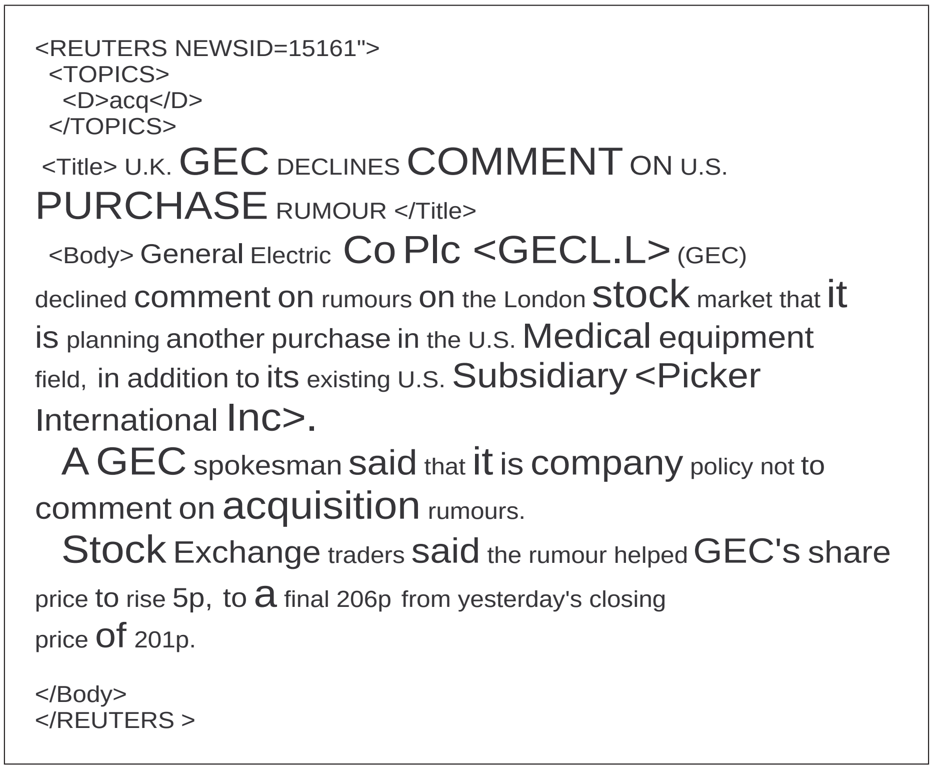
```
<REUTERS NEWSID=15161">
  <TOPICS>
    <D>acq</D>
  </TOPICS>
 <Title> U.K. GEC DECLINES COMMENT ON U.S.
PURCHASE RUMOUR </Title>
  <Body> General Electric Co Plc <GECL.L> (GEC)
declined comment on rumours on the London stock market that it
is planning another purchase in the U.S. Medical equipment
field, in addition to its existing U.S. Subsidiary <Picker
International Inc>.
    A GEC spokesman said that it is company policy not to
comment on acquisition rumours.
    Stock Exchange traders said the rumour helped GEC's share
price to rise 5p, to a final 206p from yesterday's closing
price of 201p

</Body>
</REUTERS >
```

Figure 6.11: Reuters document belonging to 'acq' class highlighted against the 'acq' class using a character-based ASTC of depth 1 using the probabilistic scoring function (pASTC-C1). Characters are scored and then scores are smoothed within word boundaries such that each word is assigned the sum of the score for its individual characters. The resulting word scores are then split into 10 partitions with the top 4 highlighted in four ascending text sizes.

indicative of a class, but surprisingly, it does identify some – as shown in Figure 6.11. Key terms, such as 'acquisition' and 'purchase' (in the title) are significantly highlighted – though the second occurrence of purchase is missed, which may well suggest that much of the highlighting in this document is subject to a degree of randomness. Certainly, while some key terms are highlighted, others, such as 'subsidiary' and 'rumour', are completely missed.

Increasing the depth to 2 has quite a dramatic effect (Figure 6.12). The highlighting already looks quite close to the ones we have seen (at depths of 8), but terms that are harder to catch, such as 'rumour' have been missed and there seems to be no sensitivity to context; for example, both occurrences of "stock" have been apportioned exactly the same weight.

However, a depth of 4 (Figure 6.13), is enough for the classifier to identify terms such as 'rumour',



```
<REUTERS NEWSID=15161">
  <TOPICS>
    <D>acq</D>
  </TOPICS>
 <Title> U.K. GEC DECLINES COMMENT ON U.S.
PURCHASE RUMOUR </Title>
  <Body> General Electric Co Plc <GECL.L> (GEC)
declined comment on rumours on the London stock market that it
is planning another purchase in the U.S. Medical equipment
field, in addition to its existing U.S. Subsidiary <Picker
International Inc>.
    A GEC spokesman said that it is company policy not to
comment on acquisition rumours.
    Stock Exchange traders said the rumour helped GEC's share
price to rise 5p, to a final 206p from yesterday's closing
price of 201p.

</Body>
</REUTERS >
```

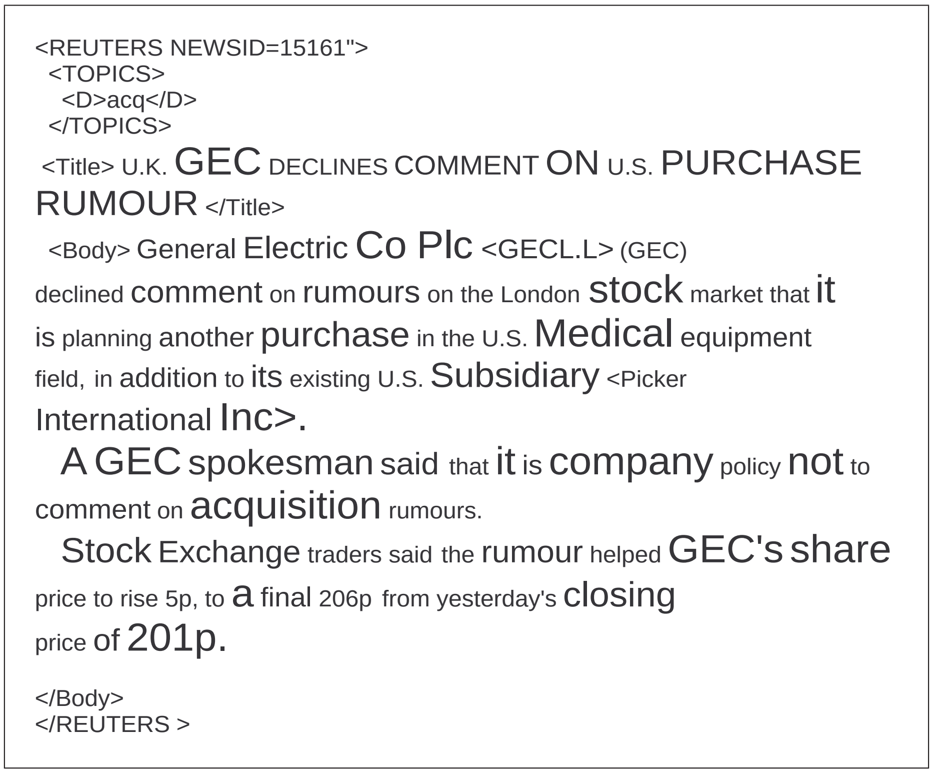
Figure 6.12: Reuters document belonging to ‘acq’ class highlighted against the ‘acq’ class using a character-based ASTC of depth 2 using the probabilistic scoring function (pASTC-C2). Characters are scored and then scores are smoothed within word boundaries such that each word is assigned the sum of the score for its individual characters. The resulting word scores are then split into 10 partitions with the top 4 highlighted in four ascending text sizes.

but comparing it to the earlier example at a depth of 8 (Figure 6.3), we can see that the deeper AST classifier is matching across terms much better and thereby catching indicative phrases such as ‘acquisition rumours’.

6.3 Conclusion

This chapter has presented a method that aims at providing the user with a view into the innards of a classification decision. This is done by highlighting the features of a document which contribute most to its overall score against a particular class (what we call, *the relative class*).

The method allows the user to evaluate a document relative to a particular class to see which terms



<REUTERS NEWSID=15161">
<TOPICS>
<D>acq</D>
</TOPICS>
<Title> U.K. GEC DECLINES COMMENT ON U.S. PURCHASE RUMOUR </Title>
<Body> General Electric Co Plc <GECL.L> (GEC) declined comment on rumours on the London stock market that it is planning another purchase in the U.S. Medical equipment field, in addition to its existing U.S. Subsidiary <Picker International Inc>.
A GEC spokesman said that it is company policy not to comment on acquisition rumours.
Stock Exchange traders said the rumour helped GEC's share price to rise 5p, to a final 206p from yesterday's closing price of 201p.
</Body>
</REUTERS >

Figure 6.13: Reuters document belonging to 'acq' class highlighted against the 'acq' class using a character-based ASTC of depth 4 using the probabilistic scoring function (pASTC-C4). Characters are scored and then scores are smoothed within word boundaries such that each word is assigned the sum of the score for its individual characters. The resulting word scores are then split into 10 partitions with the top 4 highlighted in four ascending text sizes.

in the document are relevant to that class, regardless of whether or not the class is one of the true classes of the document. The intention of the method is that when the comparison is made against a predicted class, a user is able to see a visualisation of the contribution made by each of the terms towards the classification decision, and thereby evaluate the decision of the classifier.

Using the technique, we can visualise the effect of changing certain classifier parameters such as the term-base (i.e. characters or words) or the depth. We have seen that both word-based and character-based classifiers generally identify the same words as the most significant (once the character-based significant terms are smoothed within word boundaries), but that the character-based classifier demonstrates more robustness because it is able to match partial terms and make use of non-word features such as punctuation and document formatting. These are points that we have throughout this thesis argued as an advantage of the character-based approach, but the technique of document-to-class highlighting presented in this chapter is able to visualise this. In a similar way, we have seen that the AST model of text is able to place emphasis on terms based on their context by varying the depth at which term significance is assessed. At a depth of one, no context is considered; and as we increase the depth, context becomes more important. We have seen how the significant terms change with increasing depth, but eventually seem to stabalise at and above some maximal depth; this observation correlates well with the classification results we obtained in earlier chapters.

We saw that the method is also able to give us certain clues about terms which may be relevant to a particular class - as we saw in the case of the terms "rumor" and "rumour". This further allowed to some extent the evaluation or discovery of terms that are important to a particular class. However, this is not such an important aspect of the method because such things may be done better by using more comprehensive methods which directly extract key terms from the class profile rather than by observations made on a single document.

One aspect of scoring and classification we have not explored in this section is that of the effect of the significance function on the highlighting of a document. The reason we have omitted this is that we in fact found that, on the whole, the classifier marked roughly the same terms as significant regardless of whether we used the probabilistic of root significance functions for scoring. However, although we have not explored it, the result is in fact an interesting one: that both significance functions highlight the same terms suggests that it is not the term scores against a particular class that account for the inaccuracy of the root significance function when applied to a word-based classifier, but the term scores across all the classes. In other words, it is not the ordering of the terms by score which leads to different classifications by different significance functions, but the total document score (which is nothing more than the sum of all the term scores).

This last point highlights an important limitation of the classification feature visualisation method as presented in this chapter: the resulting highlighting tells us nothing about the overall score for the document across all classes, it tells us only the importance of the terms relative to each other and the relative class. Hence the method can tell only which terms in the document are most relevant to the relative class and not which class the document is likely to be classified in.

However, the basic idea of highlighting terms can be extended in many different ways. For example, we might evaluate the terms against all classes then highlight those terms relative to a particular class only if they score above a threshold which is set across all classes. In such a framework, we would highlight only terms which scored highest over all the classes and hence we would expect more highlighting to be apparent when the class is compared against its true class than when it is compared against a false class. The disadvantage of such a method would be that we would expect to have very few terms highlighted when the document is compared against a class to which it does not belong, whereas in some domains, we may well wish to obtain some idea of which terms in the document do pertain to class, even if that class is not one of the true classes - as in the case of, for example, expertise location: here we may wish to see the terms that are relevant to a best available candidate expert (class) even if that person's expertise is not ideal.

Overall, the document-to-class feature visualisation method we have described has distinct advantages in certain domains and was originally designed with the domain of expertise location in mind, but it may also be valuable in other areas. For example, suppose the classifier were used to automatically file documents on behalf of a user within a large corporate taxonomy. If the user were unsure about where the document should be classified, the system would be able to make a number of suggestion and the user could then see the highlighting of the document relative to each of the suggested classes before making a final decision or to ensure that the classifier was making the classification decision on the correct terms in the document. Moreover, the method could be adapted and extended to serve other similar purposes.

Chapter 7

Conclusion

This thesis has developed and presented a method for classification and feature visualisation based around the suffix tree data structure.

We have argued that as a text classifier, the method has a number of strengths. Firstly, it allows the consideration of terms within a context and not simply in isolation; as such, it is able to treat text as a sequence-of-terms rather than a bag-of-terms. Secondly, the method provides a uniform approach to both character-level and word-level text modelling. Thirdly, by varying the depth of the suffix tree we use, we are able to vary the complexity of the classifier to fit the complexity of the domain we are attempting to model. Finally, the framework of the scoring method – what we refer to as the ***overlap mass score*** (OMS) – allows us to alter the way in which we interpret the significance of particular terms and term sequences, thus adding to the flexibility of the classifier framework. Each of these arguments have been explored and defended on the basis of experimental results using benchmark and specially collected data sets, with the results being compared against alternative state-of-the-art methods as they are reported in the literature.

We have also argued that the character-based approach to text classification can be just as effective as word-based approaches and additionally affords distinct advantages of its own. We have shown in Chapter 4 that the character-based approach can circumvent the obfuscations used to disguise spam email messages where many word-based methods might be fooled. We have also shown in Chapter 5, that the character-based method can compete with traditional word-based methods without the need for special pre-processing of the text; we have also argued that the character-based classifier can capitalise on features such as formatting and punctuation, which are ordinarily missed by a word-based classifier and we have shown visualisations of this in Chapter 6.

It is also worth noting that the character-based AST will tend to have a constant branching factor, limited by the alphabet size, whereas the branching factor for word-based ATSs may be very large (up to the size of the dictionary). Hence, it is usual that word-based ASTs of a particular depth will consume more resources than character-based ASTs of the same depth; which is, of course, balanced by the better performance of word-based AST classifiers at lower depths.

We have demonstrated that the classification feature visualisation method of document-to-class highlighting can give a user some indication of the features within a document that contribute to its score relative to a particular class and that such visualisations can provide clues towards the key term features of a particular class. We have argued that this can be of particular use within certain domains such as the domain of expertise location in which the sponsors of this work are interested.

The overlap mass scoring (OMS) method has been developed and a number of normalisation coefficients and significance functions have been developed and tested. In particular, we have developed two significance functions which show demonstrable success. The *root* significance function has shown success when used in conjunction with a character-based classifier on both spam data and on the Reuters data set. However, we found that when applied to a word-based classifier, performance dropped dramatically; we have so far not been able to explain why this occurs, and as such, this would certainly be a topic for future work. By contrast, the probabilistic significance function has been shown to be highly effective when used with both character- and word-based classifiers.

Unfortunately, we found that most of the methods of normalisation that we tested were ineffective, but that one, *match permutation normalisation* (Section 4.3.1), was apparently able to stabilise the classification performance. We did not continue this line of investigation, but further work may be able to develop the intuitions which initially inspired these approaches to normalisation, so that the method might achieve further improvements.

The suffix tree data structure is obviously central to the methods developed in this thesis, but it is not essential. In essence, the methods described in this thesis, evaluate the importance, or *significance*, of sequences of text (character or word) which are common between a query text and a class. Such common ‘substrings’ may be evaluated without using a suffix tree, as is done in other approaches such as, for example, that of Lodhi et al. [33]. However, the suffix tree is well suited to the modelling of text because it naturally models data of a sequential character. As a result we are better able to represent the *overlap* between a text and a class and evaluate its significance in simpler, more efficient ways than the alternative methods reported in the literature. Indeed, as we saw in Section 3.1.2, representing the task and the solution in the framework of the annotated suffix tree model of text greatly simplifies even our own approach and in many ways inspires it. Hence the annotated suffix tree is at the heart of this thesis.

7.1 Future Work

The work described in this thesis represents the initial investigation of methods of classification based around the annotated suffix tree model of text. The work could be developed further in two major ways: firstly in the refinement and extension of the model, and secondly in further testing of the model and the broadening of its scope of application to other domains.

Feature Selection As we have previously noted, we experimented with a number of feature selection methods but found none of them to improve the performance of the classifier. However, we did find some indication that the tree could be significantly pruned with relatively little impact on performance. That work is not part of the current thesis because it is not yet mature enough to be published, but it would certainly constitute a worthwhile ongoing line for future investigation. Pruning the tree would obviously have benefits in terms of computational efficiency, both in terms of memory and speed.

Extensions to the model The annotations on the suffix tree model could carry information in addition to frequency of occurrence. For example, part-of-speech tags (for a word-based classifier) or end-of-word markers (for a character-based classifier). These markers could then be used to adjust the significance associated with a particular term-sequence match.

Combinations with other methods As we have said, the methods described in this work have similarities with the methods described in Lodhi et al. [33], who developed a substring kernel for use in support vector machines. Certainly there is the possibility if integrating some of the ideas motivating that work into our work. For example, non-contiguous substring matching could be modelled within the annotated suffix tree model by adding to each node a reflexive edge; that is, a branch from each node which loops round to the same node. Hence, a frequency, probability or weight could be associated with the non-matching term; such a non-matching term could even carry a negative overlap mass score. Of course, adding this feature to the topology of the annotated suffix tree may have implications to the scoring method which would need to be investigated. Finally, carrying the amalgamation of these methods even further, the annotated suffix tree described in this thesis could be used as the basis for a more effective substring kernel for use in a support vector machines.

Experimentation on further data sets. The method has been applied in two different domains and on data sets with very different characteristics. There were the spam data sets, which favoured a character-based approach and the Reuters data set which challenged it. But there is room for testing on numerous other data sets within the text classification domain.

Computational Requirements We have given some indication of the requirements of the method in Section 4.5.2, but we have not directly addressed how the requirements might be reduced. We have explained that one important way of addressing the space requirements is simply to limit the depth of the tree. This has the dual effect of both reducing the space requirements of the classifier and improving the speed of computation (because smaller tree profiles are more easily traversed). However, such a reduction in the depth of the tree can have a negative effect on performance, so it is better to limit depth only to ovoid overtraining.

Suffix trees are known to be a maximally efficient data structure for storing all the substrings in a text, so if we want to store all substrings, we cannot reduced the memory requirements. However, we can write the suffix tree to permanent memory and retrieve only the part of the profile tree that is needed to evaluate the overlap with the query tree. We have not addressed this practical consideration at all in this work, but if the methods described here are to be implemented in a real world solution, the task of writing the profiles to memory must be tackled.

On the other hand, the speed requirements of the classifier can certainly be reduced. We currently implement a naive suffix tree construction algorithm, both for the profile trees and for the query trees. Using one of the known linear construction time algorithms would have a significant effect on the time performance of the classifier. Again, this is of primary concern if the classifier were to be implemented into a time-critical real-world system.

Application to Other Domains. Whatever the theoretical developments of the method, it would certainly be worthwhile testing the method experimentally against other data sets. Indeed it may be possible to extend the method to domains other than text classification. In theory there is no reason that the methods described in this work could not be applied more broadly, in any domain in which sequential events need to be modelled. With this in mind, the possibilities for future work are extensive.

Bibliography

[1] I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, G. Paliouras, and C.D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In V. Moustakis G. Potamias and M. van Someren, editors, *Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000)*, pages 9–17, Barcelona, Spain, 2000.

[2] apache.org. The apache spamassassin project. Webpage (last accessed November 3, 2005): `http://spamassassin.apache.org/index.html`, 2005.

[3] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, May 1999.

[4] Gill Bejerano and Golan Yona. Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics*, 17(1):23–43, 2001.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[6] Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. Word sequence kernels. *J. Mach. Learn. Res.*, 3:1059–1082, 2003.

[7] William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US, 1994.

[8] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman, 2002.

[9] Hung Chim and Xiaotie Deng. A new suffix tree similarity measure for document clustering. In *Proceedings of the 16th international conference on World Wide Web, WWW 2007*, Banff, Alberta, Canada, May 2007.

[10] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 233–240, New York, NY, USA, 2006. ACM.

[11] S. de Freitas and M. Levene. Spam on the internet: Is it here to stay or can it be eradicated? *JISC Technology and Standards Watch Reports*, (04-01), 2004.

[12] Franca Debole and Fabrizio Sebastiani. An analysis of the relative hardness of reuters-21578 subsets: Research articles. *J. Am. Soc. Inf. Sci. Technol.*, 56(6):584–596, 2005.

[13] Yan-Shi Dong and Ke-Song Han. Boosting svm classifiers by ensemble. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1072–1073, New York, NY, USA, 2005. ACM Press.

[14] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley-Interscience, New York, 1973.

[15] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, New York, NY, USA, 1998. ACM Press.

[16] T. Fawcett. Roc graphs: Notes and practical considerations for researchers, 2004. Available online: `citeseer.ist.psu.edu/fawcett04roc.html`.

[17] Peter Flach and Nicolas Lachiche. Naive Bayes Classification of Structured Data. *Machine Learning*, 57(3):233–269, 2004.

[18] Robert Giegerich and Stefan Kurtz. From Ukkonen to McCreight and Weiner: A unifying view of linear-time suffix tree construction. *Algorithmica*, 19(3):331–353, 1997.

[19] John Graham-Cummings. Spammers compendium. Webpage (last accessed October 20, 2004): `http://www.jgc.org/tsc/`, 2004.

[20] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge Unversity Press, 1997.

[21] Louise Guthrie, Elbert Walker, and Joe Guthrie. Document classification by machine: theory and practice. In *Proceedings of the 15th conference on Computational linguistics*, pages 1059–1063, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

[22] Y. Hassan-Montero and V. Herrero-Solana. Improving tag-clouds as visual information retrieval interfaces. In *Proceedings of the International Conference on Multidisciplinary Information Sciences & Technologies*, 2006.

[23] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[24] F. Johannes. A study using n-gram features for text categorization, 1998.

[25] Chunyu Kit and Yorick Wilks. The virtual corpus approach to deriving ngram statistics from large scale corpora. In C. N. Huang, editor, *Proceedings of International Conference on Chinese Information Processing Conference*, Beijing, 1998.

[26] Stefan Kurtz. Reducing the space requirement of suffix trees. *Software Practice and Experience*, 29(13):1149–1171, 1999.

[27] N. Lachiche and P.A. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using roc curves. In *Proc. 20th International Conference on Machine Learning (ICML'03)*, pages 416–423. AAAI Press, January 2003.

[28] Honglak Lee and Andrew Y. Ng. Spam deobfuscation using a hidden markov model. In *CEAS*, 2005.

[29] David D. Lewis. Reuters-21578 distribution 1.0. Webpage : `http://www.daviddlewis.com/resources/testcollections/reuters21578/`, 1997.

[30] David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15. Springer Verlag, Heidelberg, 1998.

[31] David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 298–306, Zürich, CH, 1996. ACM Press, New York, US.

[32] Y. H. Li and Anil K. Jain. Classification of text documents. *Comput. J.*, 41(8):537–546, 1998.

[33] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444, 2002.

[34] Bingwen Lu and Ting Chen. A suffix tree approach to the interpretation of tandem mass spectra: applications to peptides of non-specific digestion and post-translational modifications. *Bioinformatics*, 1990(02):113ii–121, 2003.

[35] C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[36] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification, 1998.

[37] Edward M. McCreight. A space-economical suffix tree construction algorithm. *J. ACM*, 23(2):262–272, 1976.

[38] T.A Meyer and B Whateley. Spambayes: Effective open-source, bayesian based, email classification system. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, July 2004. Available online: `http://www.ceas.cc/papers-2004/136.pdf`.

[39] Eirinaios Michelakis, Ion Androutsopoulos, Georgios Paliouras, George Sakkis, and Panagiotis Stamatopoulos. Filtron: A learning-based anti-spam filter. In *Proceedings of the First Conference*

on Email and Anti-Spam (CEAS), Mountain View, CA, July 2004. Available online: `http://www.ceas.cc/papers-2004/142.pdf`.

[40] B. Mirkin. *Mathematical Classification and Clustering*. Kluwer Academic, Dordrecht, 1996.

[41] Boris Mirkin. *Clustering For Data Mining: A Data Recovery Approach (Chapman & Hall/Crc Computer Science)*. Chapman & Hall/CRC, 2005.

[42] Rajesh Pampapathi, Boris Mirkin, and Mark Levene. A suffix tree approach to text categorisation applied to email filtering. In *Proc. UK Workshop on Computational Intelligence (UKCI2005)*, pages 212–219, London, September 2005. Available online: `http://www.dcs.bbk.ac.uk/ukci05/ukci05proceedings.pdf`.

[43] Rajesh Pampapathi, Boris Mirkin, and Mark Levene. A suffix tree approach to anti-spam email filtering. *Mach. Learn.*, 65(1):309–338, 2006.

[44] F. Peng and D. Schuurmans. Combining naive bayes and n-gram language models for text classification, 2003. submitted to The 25th European Conference on Information Retrieval Research (ECIR).

[45] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[46] M. F. Porter. An algorithm for suffix stripping. In *Readings in information retrieval*, pages 313–316. Morgan Kaufmann Publishers Inc., 1997.

[47] Vijay Raghavan, Peter Bollmann, and Gwang S. Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Trans. Inf. Syst.*, 7(3):205–229, 1989.

[48] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automated Document Processing*, pages 313–323. Prentice-Hall, Inc, 1971.

[49] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05. Available online: `http://citeseer.ist.psu.edu/sahami98bayesian.html`.

[50] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[51] Karl-Michael Schneider. A comparison of event models for naive bayes anti-spam e-mail filtering. In *Proc. 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003)*, pages 307–314, Budapest, Hungary, 2003.

[52] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[53] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[54] Noam Slonim, Gill Bejerano, Shai Fine, and Naftali Tishby. Discriminative feature selection via multiclass variable memory markov model. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 578–585, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[55] David Surkov. *Inductive Confidence Machine for Pattern Recognition: is it the next step towards AI?* PhD thesis, Royal Holloway University of London, 2004.

[56] E. Ukkonen. Constructing suffix-trees on-line in linear time. *Algorithms, Software, Architecture: Information Processing*, 1(92):484–92, 1992.

[57] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory (Information Science and Statistics)*. Springer, November 1999.

[58] Peter Weiner. Linear pattern matching algorithms. In *FOCS*, pages 1–11. IEEE, 1973.

[59] S. M. Weiss, N. Indurkhya, T. Zhang, and F. J. Damerau. *Text Mining: Predictive Methods for Analyzing Unstructured Information*. Springer, 2005.

[60] Gregory L. Wittel and S. Felix Wu. On attacking statistical spam filters. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, July 2004. Available online: `http://www.ceas.cc/papers-2004/170.pdf`.

[61] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.

[62] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR ’01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, New York, NY, USA, 2001. ACM Press.

[63] Jian Zhang and Yiming Yang. Robustness of regularized linear classification methods in text categorization. In *SIGIR ’03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 190–197, New York, NY, USA, 2003. ACM Press.

[64] Le Zhang, Jingbo Zhu, and Tianshun Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(4):243–269, 2004.