(12) UK Patent Application (19) GB (11) 2 395 301 (13) A

(43) Date of A Publication 19.05.2004

(21) Application No: 0226145.1

(22) Date of Filing: 08.11.2002

(71) Applicant(s):
Magus Research Ltd
(Incorporated in the United Kingdom)
Rowlandson House, 289-293 Ballards Lane,
LONDON, N12 8NP, United Kingdom

(72) Inventor(s):
Barry David Otley Adams

(74) Agent and/or Address for Service:
Venner Shipley & Co
20 Little Britain, LONDON, EC1A 7DH,
United Kingdom

(51) INT CL^7:
G06F 17/00

(52) UK CL (Edition W):
G4A AUDB

(56) Documents Cited:
WO 2002/065427 A2 **US 6388592 B1**
US 6182058 B1 **US 20020091676 A1**

(58) Field of Search:
UK CL (Edition V) **G4A** AUDB AUXX
INT CL^7 **G06F** 17/00
Other: **ONLINE:EPOQUE, INTERNET**

(54) Abstract Title: **A method of classifying a set of data**

(57) A two-stage method of classifying a set of data comprises: in a training phase: estimating from pre-classified sets of training data the probability of at least two features occurring together in each of at least two categories; and calculating, from the appropriate estimated probability and a measure of the probability of the features occurring together in the sets of training data if the features were independent, an error factor for the two features for each category. This involves comparing a vector of the estimated probabilities of the features occurring together in each of the categories with a vector of the probabilities, if the features were independent, of the two features occurring together in each category. In a recognition phase, the error factor is used to classify the set of data. The error factor is a measure of the falsehood of the assumption of independence assumed by the Naïve Bayes algorithm. Using the error factor in the recognition phase can then correct at least partially for this falsehood, resulting in improved classification of sets of data, compared to the classic Naïve Bayes algorithm.

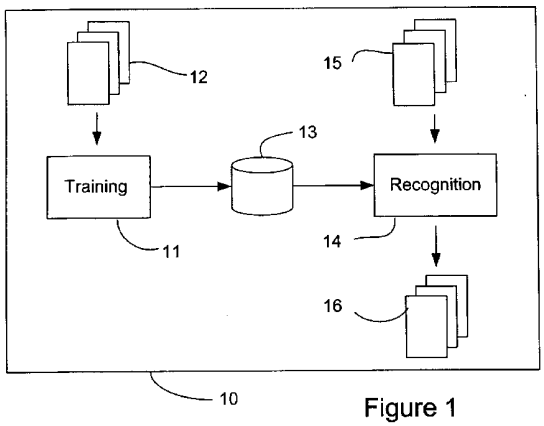

Figure 1

GB 2 395 301 A

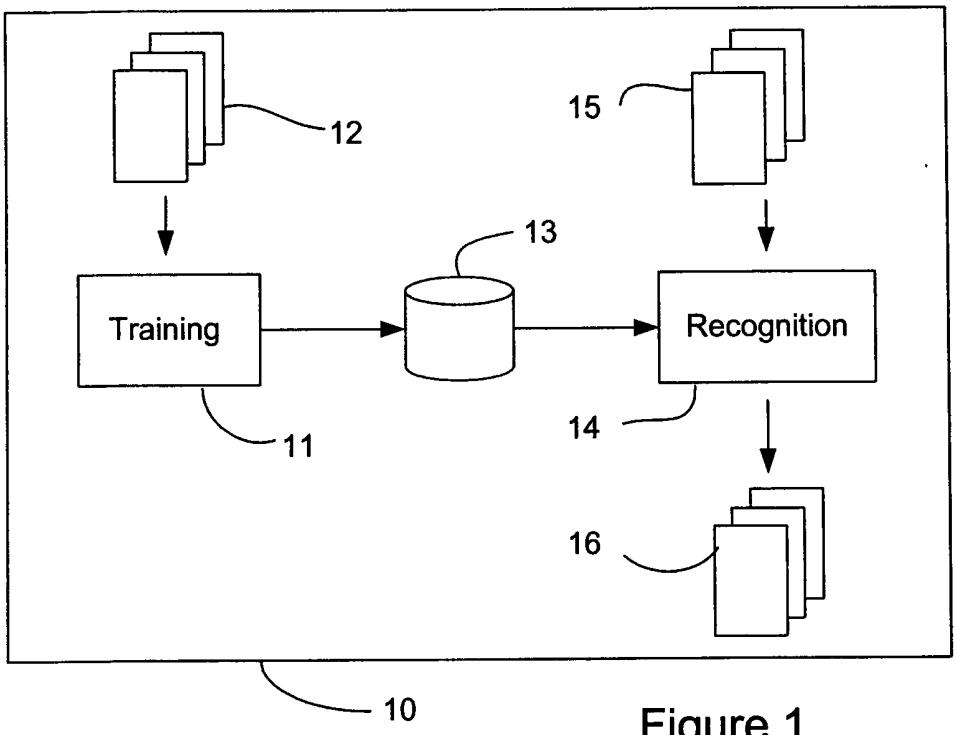

Figure 1

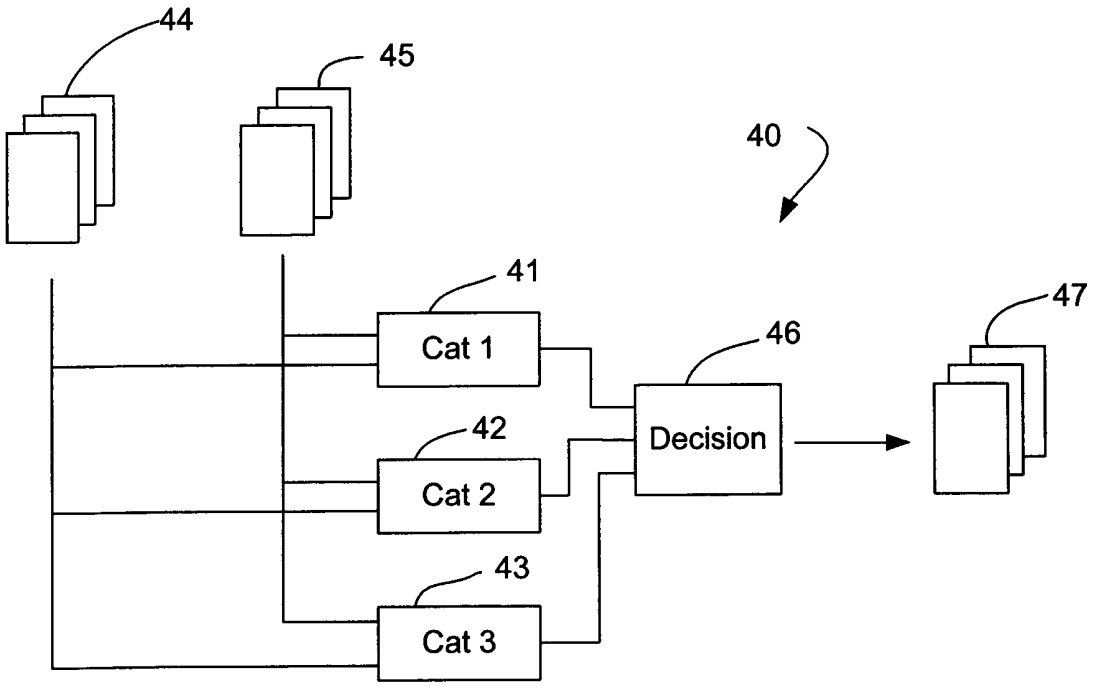

Figure 4

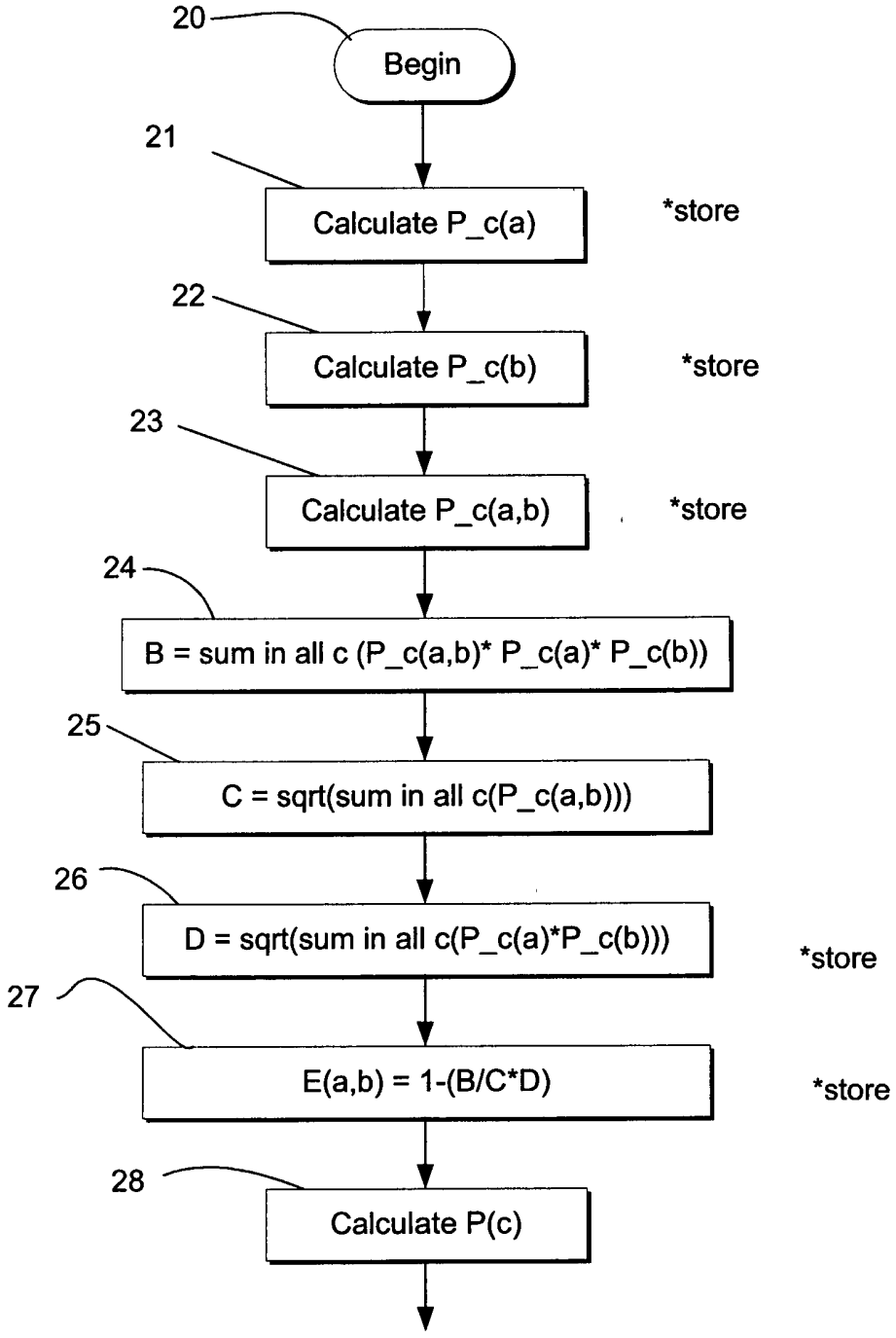
20
Begin
21
Calculate P_c(a)
*store
22
Calculate P_c(b)
*store
23
Calculate P_c(a,b)
*store
24
B = sum in all c (P_c(a,b)* P_c(a)* P_c(b))
25
C = sqrt(sum in all c(P_c(a,b)))
26
D = sqrt(sum in all c(P_c(a)*P_c(b)))
*store
27
E(a,b) = 1-(B/C*D)
*store
28
Calculate P(c)

Figure 2

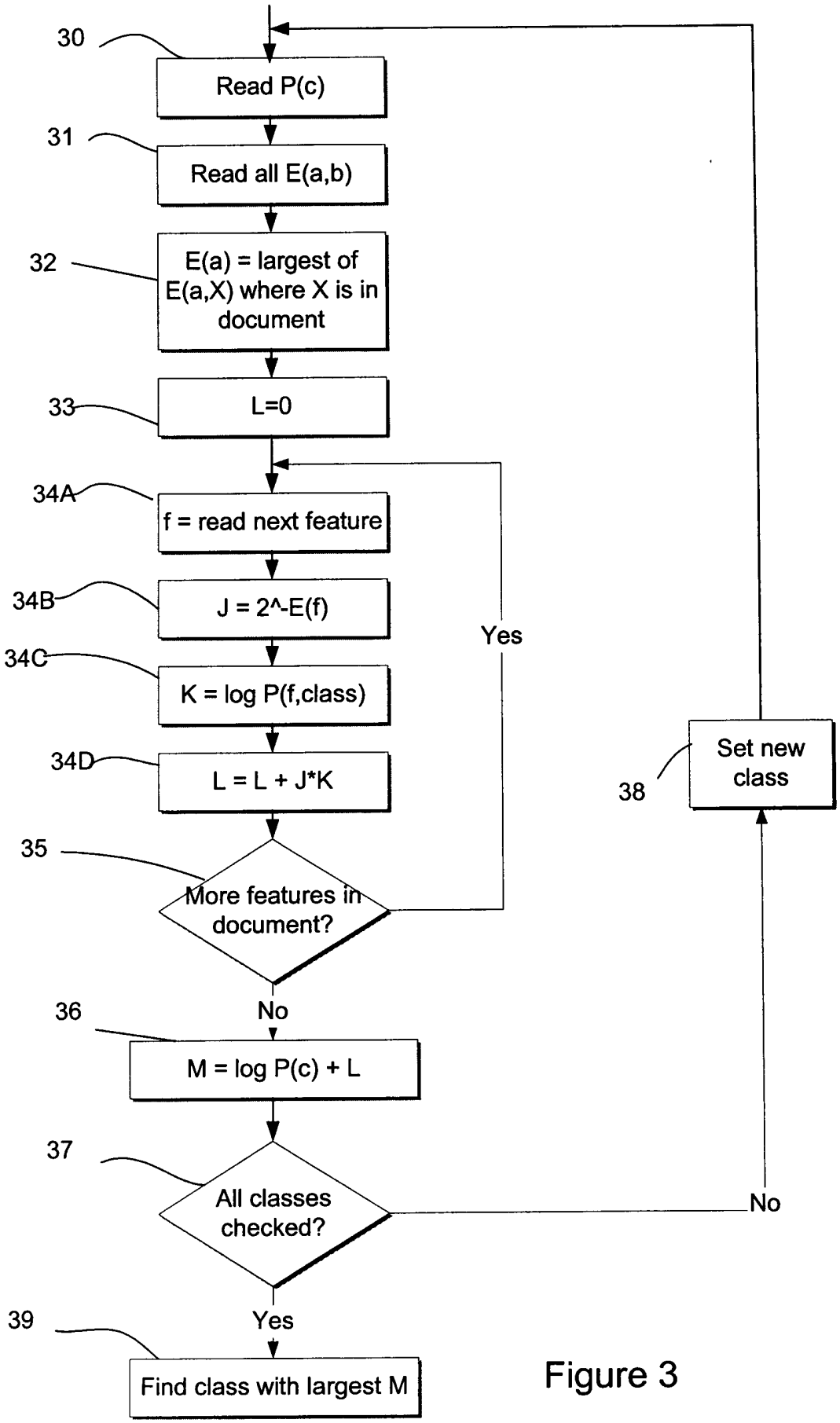
30
Read P(c)
31
Read all E(a,b)
32
E(a) = largest of E(a,X) where X is in document
33
L=0
34A
f = read next feature
34B
J = 2^-E(f)
34C
K = log P(f,class)
34D
L = L + J*K
35
More features in document?
Yes
No
36
M = log P(c) + L
37
All classes checked?
No
Yes
38
Set new class
39
Find class with largest M

Figure 3

Data Set Classification

This invention relates to a method of, and apparatus for, classifying a set of data. An error factor is calculated in a training phase, and used to classify data in a recognition phase.

A well-known algorithm for classifying sets of data is the Naïve Bayes algorithm. Data sets may be textual documents, such as newspaper articles, or documents containing textual and non-textual information, such as web pages. A computer system running a program implementing the algorithm analyses categorised documents in a training phase, and used resulting probability data to categorise uncategorized documents in a recognition phase. Such systems can be used to determine whether e-mail messages are unsolicited advertisements as part of a so-called 'spam filter', or to group together or merely label with a category web pages identified by a web search.

The Naïve Bayes Algorithm can be briefly described from its origin in probability theory. Each of the categories is labelled with an integer c, Given a document D, the probability $P(c|D)$ of the Document D being belonging to category (or class) c, is given using Bayes law of probability as:

$$P(c|D) = P(D|c)\,P(c) \;/\; P(D)$$

$P(D|c)$ is the probability of producing document D from a random set of documents in category c. $P(c)$ is the prior probability of a document picked at random being in category c.

The document is then assumed to be a collection of independent features, usually words, although pairs of words, phrases, or more complex grammatical structures can also use be used. If the features are all independent, that is the occurrence of one feature in a document does not make the occurrence of any other feature any

more or less likely, then the probability P(D) of the document is the probability of each of the features multiplied together:

$$P(D) = P(f\_1) * P(f\_2) * P(f\_3) \ldots$$

And similarly:

$$P(D \mid c) = P(f\_1 \mid c) * P(f\_2 \mid c) * P(f\_3 \mid c)..$$

Following which:

$$P(c \mid Doc) = P(f\_1 \mid c)\, P(f\_2 \mid c)\, P(f\_3 \mid c) \ldots P(c)\, / P(D)$$

Since there definitely is a document to study, P(D) = 1.

It is then simple to define a categorizing method as the process of choosing the category with the highest probability given for the given document:

$$c(D) = \text{Argmax\_c} \left[P(c) \prod_{\text{each f in D}} (f \mid c) \right] = \text{Argmax\_c} \left[\log P(c) + \sum \log P(f \mid c) \right] \quad \text{(equation 1)}$$

In the above, Argmax_c means find the value of c such that the quantity in brackets has the highest value.

It is possible to train the document classification system in advance by finding the Probabilities P(f | c) of each feature in each class using a set of training document D(c,1)..D(c, n_c) in each category c.

If the training set is a representative sample of the real life frequencies of various documents it is assumed that:

$$P(c) = N\_c \,/ \sum_{C} N\_c$$

where N_c is the number of Documents in each category

Otherwise, P(c) may be set so that all categories are, *a priori*, equally probable, which removes P(c) from equation 1 since only the differences between the different probabilities make a contribution.

Let N_f_c be the total number of occurrences of the feature f, in the training documents of class c. Then:

$$N\_c = \sum_f N\_f\_c$$

Here, N_c is seen to be the total number of occurrences of any features in the training documents for class c. For the class c we have,

$$P(f|c) \sim N\_f\_c / N\_c$$

i.e. the total probability of the feature f, given that the class is c, is the number of occurrences of that feature in all the training documents for the class c, divided by the total number of occurrences of any feature in class c.

Because in practice the amount of training data is finite, the probability of a feature occurrence is quantised. Accordingly, a rare feature may show $P(f|c) = 0$, despite having a finite probability. Such rare features would be given too much weight in the algorithm if they occur in the data to be recognised but not in the training data, so it works better in practice to have P(f|c) instead defined as:

$$P(f|c) \sim (1+N\_f\_c) / N\_c \qquad \text{(equation 2)}$$

The algorithm can be summarised thus:

Training: find the number of occurrences of each feature in the training set for each category, then compute and store the value P(f|c) for each feature and category using equation 2.

Recognition of document: Divide the document into features, and using the stored P(f|c)'s from the training phase, find the category of the document using equation 1.

It is considered that the above Naïve Bayes algorithm is wholly satisfactory when the features of documents are independent. However, this is rarely true in practise, so the algorithm is sub-optional in real-life situations. It is an aim of the invention to address the deficiencies of the Naïve Bayes algorithm.

According to a first aspect of the invention, there is provided a method of classifying a set of data, the method comprising: in a training phase: estimating from pre-classified sets of training data the probability of at least two features occurring together in each of at least two categories; and calculating, from the appropriate estimated probability and a measure of the probability of the features occurring together in the sets of training data if the features were independent, an error factor for the two features for each category; and, in a recognition phase: using the error factor to classify the set of data.

Using this method, it is possible to draw from training sets of data an error factor which is a measure of the falsehood of the assumption of independence assumed by the Naïve Bayes algorithm. Using the error factor in the recognition phase can then correct at least partially for this falsehood, resulting in improved classification of sets of data, compared to the classic Naïve Bayes algorithm, when the assumption of the independence of features is not true.

Probabilities may be estimated or calculated in any convenient manner. In the embodiments, the probability of a feature occurring (or features occurring together) is calculated from a count of the number of sets of data in each category in which that feature occurred (or those features occurred together) and a count of the

number of sets of data in that category. However, less processor intensive techniques may be used instead.

Preferably the calculating step comprises comparing a vector of the estimated probabilities of the at least two features occurring together in each of the categories with a vector of the probabilities, if the features were independent, of the two features occurring together in each category. This produces a measure of the degree of falsehood of the assumption of independence which is considered to be a good measure and which is relatively simple to calculate, given modern programming languages, compilers and data processors. The comparing step may comprise dividing the dot product of the two vectors by the product of the moduluses of the two vectors.

The using step may include finding the product (or some other similar function) of a function of the error factor and a measure (which in the embodiments is the logarithm) of the estimated probability of the two features occurring together. This is advantageous since it is desired for processing to be quick in the recognition stage (to allow classification of multiple seta of data quickly) and the calculations provide a reliable indication of the class to which the set of data is best suited and which, depending on the function of the error factor used, is relatively quick to calculate.

The function of the error factor preferably involves finding a power of a value, the power including a measure of the error factor. In the embodiment, the function of the error factor is 2 to the power of minus the error factor, which is particularly easy to calculate and which has been found to result in particularly good classification of sets of data.

The using step may include summing for at least two feature combinations in a set of data the products (or other similar function) of the function of the error factor for that feature combination with the measure (which in the embodiments is the logarithm) of the estimated probability for that feature combination. This allows a numerical value to be provided for a set of data which value takes into account plural combinations of features and which gives a measure of the similarity of the data with other sets of data in a given category.

The using step preferably include summing a measure (which may be a logarithm) of the probability of a set of data falling into the category being tested. This feature can provide compensation for errors which might occur should there not be equal numbers of training sets of data in each of the categories.

The using step preferably includes finding the category for which a largest value is calculated, allowing the best category to be found with a relatively simple calculation.

According to a second aspect of the invention, there is provided apparatus for classifying a set of data, the apparatus comprising: an estimator arranged in a training phase for estimating from pre-classified sets of training data the probability of at least two features occurring together in each of at least two categories; a calculator arranged in the training phase for calculating, from the appropriate estimated probability and a measure of the probability of the features occurring together in the sets of training data if the features were independent, an error factor for each category; and a classifier arranged, in a recognition phase, to use the error factor to classify the set of data.

Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings, of which:-

Figures 1 and 4 illustrate systems according to one aspect of the invention and operating according to another aspect of the invention; and
Figures 2 and 3 illustrate operations performed by a training program and a recognition program, respectively, which together form part of the Figures 1 and 4 systems.

A classification system according to the invention is shown in Figure 1. Referring to Figure 1, the system 10 comprises a training program module 11, which is arranged to receive pre-classified training documents 12, and to provide probability data and error factor data to a hard disk 13 for storage. A recognition program

module 14 is arranged to use the data stored on the hard disk 13 in the classification of documents 15 which are not pre-classified, and to provide documents 16 labelled with their class so determined at an output 17. The system may be implemented on a general purpose computer, or it may be distributed on networked computers for example. In most circumstances, the modules 11 and 14 will not be operational simultaneously.

Taking two features 'a' and 'b', if the probability is independent, then:

$$P(a,b) = P(a)P(b),$$

Otherwise:

$$P(a,b) \neq P(a)P(b)$$

For example, if the two features a and b always occur together and never occur separately, then:

$$P(a,b) = P(a) = P(b)$$

From these equations, it can be seen that if two features always occur together in a particular category, the importance of the two features is overstated by the Naïve Bayes algorithm. Worse still, the two features a and b may occur together more often in a category which is different to the categories that each feature occurs in alone most often. Thus, equation (1) may mis-predict the category of the document.

In the training data, the probability of the two words occurring together in each class P_c(a,b) could be estimated thus:

P_c(a,b) = N_a&b_c in document / N_c(N_c-1)

Where N_a&b_c is the number of times both words a and b occur in any of the documents in Class c.

The probability given the assumption of independence is:

$$P\_c(a)P\_c(b) = N\_a\_c * N\_b\_c / N\_c * N\_c$$

where N_a_c is the number of times 'a' word occurs in any of the documents in category c, and N_b_c is the number of times word 'b' so occurs.

These values can be treated as vectors in a space with one dimension for each category, and the correlation between the probability in the assumed case of independence and the actual case where the words are not necessarily independent can be found. The Error factor caused by the assumption of independence for the two features using the dot product between the two vectors of probability can be expressed thus:

$$E(a,b) = 1 - \frac{\underline{P}(a,b) \cdot \underline{P}(a)\underline{P}(b)}{|\underline{P}(a,b)| \; |\underline{P}(a)\underline{P}(b)|} \qquad \text{(equation 3)}$$

or expressed another way:

$$E(a,b) = 1 - \frac{\sum_c P\_c(a\ b)\ P\_c(a)P\_c(b)}{\text{sqrt}\left(\sum_c P\_c(a\ b)^2\right) \text{sqrt}\left(\sum_c P\_c(a)P\_c(b)^2\right)}$$

Error factors calculated using equation 3 are stored at the end of the training process, along with each of the probabilities for P(f|c) given by equation 2. The value of E(a,b) is zero if the features are independent in the training data and tends toward 1 if the two features together are found in a category different to the categories where each is found alone. E(a,a) is defined as zero.

In the recognition stage, the Error factor of the feature 'a' alone is estimated as the maximum value of E(a,b) for all the b that occur in the document:

E(a) = Max (E(a,b) | where b occurs in the document),

Using this error factor, an improved equation 1 can be written thus:

$$c(D) = \text{Argmax\_c} \left[\log P(c) + \sum_{\text{each f in Doc}} g(E(f)) \log P(f|c) \right] \quad \text{(equation 4)}$$

Where g(x) is some monotonically decreasing function of x. It is required that g(0) =1, so that equation 1 is recovered when all the error factors are zero, i.e. when the assumption of independence is true, the Naïve Bayes algorithm is expected to operate optimally.

It is then assumed that the form of g(x) is g(x) = exp(- k x), with unknown k. From experimentation with various values of k on real problems it has been found that the best value of k is about 0.7. It is taken into account that if two features always occur together, they should be counted together as a single feature. Thus, in this case the contribution of each of the features is needed to be one half the usual factor, thus:

g(1) = 0.5 -> exp(-k) = 0.5 , i.e. k = ln 2,

Then, g(x) = exp(- x ln(2)) = 2^-x

Thus the finished categorizer equation is,

$$c(D) = \text{Argmax\_c} \left[\log P(c) + \sum_{\text{each f in Doc}} 2^{-E(f)} \cdot \log P(f|c) \right] \quad \text{(equation 5)}$$

The embodied algorithm can be described as having a training phase in which the number of occurrences of each feature in the training sets is found for each category, and the value P(f|c) for each feature and category is computed using equation 2 and stored. Then, the number of occurrences of pairs of features occurring anywhere in a document is found and the values E(a,b) for each pair of features is calculated using equation 3, and the results are stored.

In a recognition phase, the document is divided into features. For each feature, 'a', in the document its error factor E(a), is calculated as the maximum value of E(a,b) for all the other features 'b' in the document. Then, equation 5 is used to find the category of the document, making use of the stored P(f|c) values.

As explained, the embodied algorithm is divided into two parts. In the training phase, a set of stored values is built by a first program from a set of documents each labelled with a particular category. In the recognition phase, another program determines a category for an unlabelled document using the set of stored values.

A program implementing the training phase of the algorithm is hereafter termed the training program. It has input as its a set of labelled (pre-categorised) documents, and as its output a set of arrays of numbers to be stored on the computer.

Firstly, the pre-categorised training documents are read in, and from this the number of categories and the number of training documents is determined. For simplicity, it is assumed in this example that the documents are textual, and the features examined are words of the text. Below is pseudocode for implementing the training program. In this pseudocode, the size of various tables used is limited by preset maximum values, a maximum number of unique words (Nmaxwords), a maximum number of categories (Nmaxcat), a maximum Number of documents (Ndocs), and a maximum length of each document. It is assumed that the programming language or its libraries contain hash tables or associative arrays, that is arrays that may be addressed by a string of text, or other complex object, rather than by a number. The below pseudocode uses hash tables as sparse arrays, in order to allow the program to be executable quickly.

The training program reads each document in turn, assigning a category number to each new category found in a document. It then reads in turn each word out of the current document, building totals of the number of words in each category and the number of occurrences of each unique word in each category. It uses a temporary hash table as a sparse array to total the occurrences of each word in a single document, and then uses this hash table to total the number of occurrences for each pair of words in a single document for each category. The total number of word pairs is kept in a hash table keyed by pairs of numbers.

Having finished reading all the training documents, the training program computes the probability of each word in each category, using the totals found above, and then computes the error factor given by equation 3. The number of occurrences of each pair of words is computed in each category. Finally, the training program finds the logarithms of the probabilities, and writes the required data to disk.

The recognition program is simpler. It first reads in the data prepared by the training program in the training phase. For each new document, the recognition program reads the words in the document, finding the total number of occurrences of each unique word in the document. It finds the maximum error factor for each word in the document given all the other words that occur in the document, and computes the probabilities for each potential category using equation 5. Finally, the program finds the category with the largest probability and labels the document with the name of this category.

The Training Program

```
Begin Training Program

# The below initialises the variables and arrays.

integer Nmaxcat = 100
integer Nmaxwords = 50000
```

```
integer Nmaxdocs = 10000
integer Ncats = 0
integer Ndocs = 0
integer Nwords =0
array docsincat = array of integers, size Nmaxcat, each value preset to zero
array wordsInCat = a 2 dimensional array  of integers, size Nmaxwords by

Nmaxcats, each value preset to zero
array catTotals = array of integer, size Nmaxcat
array catNames = array of Strings, size Nmaxcat

hashtable CatNumbers = hashtable of size Nmaxcat, mapping strings onto integers
hashtable WordsNumbers = hashtable of size Nmaxwords, mapping string onto

integers

array catpairs = array of hashtables, size Nmaxcat, each Hashtable maps a pair of

integers onto a single integer

# Loop for each document in the training set.

while( more Documents){
 String Dcat = read the category of the document from the file
 integer ic = 0

#  Below a number of the category is set, a new one being defined for each
document.

 If (already defined catNumbers[Dcat]){
   ic = catNumbers[Dcat]

#  get the number from the hash table, since we already have since a category of
```

```
# this name.

 } else {
   ic = Ncats
   catNumbers[Dcat] = Ncats
   catNames[Ncats] = Dcat
   Ncats = Ncats+1

#   Give each new category a new number, and store the name and number to be
# looked at later.

 }
 docsincat[ic] = docsincat[ic]+1

# The above counts the number of documents in each category.

 hashtable wordoccs = hashtable of words of size of Nmaxwords, mapping integers
to integers

#  The above hashtable is created empty for each document, and holds the count
#  of how many times each different word occurs in the document, this is used to
#  speed up counting the occurrences for each pair of words.

 integer j=0

# Below each word in the document is read and counted.

 while( more words in the document){
   String word = read next word from the document
   integer iw = 0

# Below, a number is found for each of the words, a new number being assigned if
#  the word has not been seen before.
```

```
    if (already defined wordsNumbers[word]){
     iw = wordsNumbers[Word]

#    If a number has already been assigned for this word, it is used.

    } else {
     iw = Nwords
     wordNumbers[word] = Nwords
     Nwords = Nwords+1

#    For a word not previously seen, a new number is assigned and stored in the
#     hashtable to be found next time it occurs.

    }
    wordsInCat[iw][ic] = wordsInCat[iw][ic]+1

#  Above is counted the total number of occurrences of each word, in each
#  category.

    catTotals[ic]=catTotals[ic]+1

#   Above is counted the number of words in the training documents for each
#  category.

    wordoccs[iw] = wordoccs[iw]+1

#   The above code counts the total number of occurrences of each word, in all the
#  documents

  }

#  Below is counted how many documents in each category contain both words.
```

```
 for(integer iw = each key of hashtable wordoccs){

#  i.e. the number of each unique word contained in the document

   for(integer jw = each key of hashtable wordoccs, and jw<iw){

#  jw and another unique word iw
#  Since P(a,b)=P(b,a), it is needed only to store pairs with jw<iw

     catpairs[ic][iw,jw] = catpairs[ic][iw,jw]+1
   }
 }

} # End for each document

# Below is found the probabilities P(f|c) from equation 2, which are stored in an
# array.

array ProbWordsInCat = a 2 dimensional array of floating point values, size Nwords
by
Ncats
for( cat =0 to Ncats-1){
 for( word=0 to Nwords-1){
  probWordsInCat[nw][cat] = (1 + wordsInCat[nw][cat])/catTotals[cat]
 }
}

# Below is found the error factors E(a,b) from equation 3, which are stored in an
# array.

array ErrorFactors = a 2 dimensional triangular array of floating point values size
Nwords by Nwords
```

```
# Since E(a,b)=E(b,a), space can be saved by keeping it in a  triangular array, i.e.
# ErrorFactor[iw][jw] need only be stored when jw<iw.

for(integer iw = 0 to Nwords-1){
 for(integer jw= 0 to iw-1){
  float sumdotprod = 0
  float sumpair2 = 0
  float sumprob2 = 0

# Since there are only three summations to perform over the categories, they are
# done  together in one loop.

  for(integer c= 0 to Ncats-1){
   float pairprob = catspairs[ic][iw,iw]/ (catTotals[ic] * (catTotals[ic]-1))
   float temp = probWordsInCat[iw][cat] * probWordsInCat[jw][cat]
   float dotprod = pairprob * temp
   sumdotprod = sumdotprod + dotprod

#    the last line of code sums the numerator in equation 3.

   sumpair2 = sumpair2 + pairprob*pairprob

#   the above sums the square of the modulus (length) of P(a,b) in equation 3.

   sumprob2 = sumprob2 + temp * temp

# the above sums the square of the modulus (length) of P(a)*P(b) in equation 3.

  }
  ErrorFactor[iw][jw] = 1 - sumdotprod / (sqrt(sumpair2)* sqrt(sumprob2) )

# This completes equation 3 and stores the number in an array.
```

```
 }
}

# The logarithm of the probabilities, needed for use in equation 5, are found
# below.

# Identical results are found whatever the base of the logarithm, but
# Natural log, base e (2.71....), is used here.

array logProbWordsInCat = a 2 dimensional array of floating point values, size
Nwords by Ncats
for( cat =0 to Ncats-1){
 for( word=0 to Nwords-1){
  logProbWordsInCat[nw][cat] = ln( probWordsInCat[nw][cat]  # i.e. take Natural
Log of each value
 }
}
writetodisk( nCats,nWords, catNames, wordNumbers, logProbWordsInCat,
ErrorFactor)

End TrainingProgram
```

The main calculation steps made by the training program are shown in Figure 2. Referring to Figure 2, after beginning at step 20, the probability of the feature a appearing in class c is calculated at step 21. The probability of feature b appearing in class c is calculated at step 22, following which the probability of both features a and b occurring is calculated at step 23. The numerator of equation 3 is calculated at step 24, and the two parts of the denominator are calculated at steps 24 and 25 respectively. The error factor is then calculated at step 27. This process is calculated for each pair of features. The probability of a document being categorised in class c is the calculated at step 28. This step involves simply dividing the number of documents classified into category c by the total number of training

documents. The results of steps 21, 22, 23, 26 and 27 are stored for later use by the recognition program.

The Recognition Program

```
# the below initialises integers and arrays.

integer nCats
integer nWords
array catNames = array of Strings to be read
hashtable wordNumbers = an Hashtable mapping Strings to numbers to be read
array probWordsInCat = a 2 dimension array of floating point values
array ErrorFactor = a 2 dimensional triangular array of floating point values

# All the data stored after the training phase is read into memory.

readfromdisk( nCats, nWords, catNames, wordNumbers, logprobWordsInCat,
ErrorFactor)
float log2 = ln 2   # Natural log of 2

# Each document is categorised, one at a time, in the loop below.

while(more Documents){

# A hashtable is built that stores the count of the occurrences of each different
# word in a document.

hashtable wordOcc = a hashtable size nWords, mapping integers to integer (i.e. a
sparse array)

# Below the document is read one word at a time, counting the occurrences of
# each of the different words in the document.
```

```
while(document has more words){
  String word = next word in the Document
  if (!defined wordNumbers[word]){ skip to next word }

# Any word seen in the training documents has been assigned a number in the
# training phase, which number is retrieved below.

  integer wn = wordNumbers[word]
  wordOcc[wn] = wordOCc[wn]+1
 }

# In the recognition stage, an estimate of the Error factor of the feature 'a'
# alone is found as the maximum value of E(a,b) for all the b that occur in the
# document. This maximum is found below, E(a) = Max (E(a,b)) where b occurs
# in the document.

array maxError = an array of floating point values of length nWords

# Each unique word in the document is looped over.

 for( integer iw = each key of the hashtable wordOcc){
  float max = 0

# Below is an inner loop over another (different) unique word in the document.

  for (integer jw = each key of hashtable wordOcc){
   if (iw == jw){ skip to next key }
   float E = 0

# Below, the fact that E(a,b)=E(b,a) is used to get the value from the stored
# triangle array.

   if (iw<jw){ E = ErrorFactor[jw][iw] } else { E = ErrorFactor[iw][jw] }
```

```
    if (E>max){ max = E }
   }
   maxError[iw] = max
  }

# Below are calculated the probabilities for the document being in each of the categories.
# This is the inner part of equations 5. If necessary, the categoriser can be biased
#   to a particular document type, by giving probCats[cat] (the probability of the
#  document being in a particular category) a suitable predetermined value.

 array probCats = an array of floating point values of size nCats, values preset to zero
 for(integer iw = each key of the hashtable wordOcc){

# The error factor for the word is used to degrade the importance of problematic
# words, as in equation 5,

   float etemp = exp( - log2 *maxError[iw] )
   for(integer cat=0 to nCats-1){

# The contributions to the probabilities of the different categories are summed
# below.

    probCats[cat] = probCats[cat] + wordOcc[iw]* etemp* logprobWordsInCat[iw][cat]
   }
 }

# If necessary, the probabilities could be normalised so that they add up to 1, or
# a 'don't know' option included if one category is determined not to be
# particularly favoured over the others.
# Below is calculated the argmax part of equation 5, i.e. the category
```

```
# with the maximum probability is found, and the document is labelled accordingly.

 integer catNumber = -1
 float max = -infinity                  # some very large negative number
 for(int i=0 to nCats-1){
  if (probCat[i]>max){
   max = probCat[i]
   catNumber = i
  }
 }

# The name of the numbered category for the array of names generated in the
#  training phase is read, and the document labelled accordingly

 String catName = catNames[catNumber]
 label the document with the String catName
} # Next document

End RecognitionProgram
```

The main calculation steps carried out by the recognition program are shown in Figure 3. Referring to Figure 3, the values of the probability of a document being classified to category c and the error factors are read from memory at steps 30 and 31 respectively. At step 32, the error factor for a feature 'a' alone is estimated as the maximum value of E(a,b) where the other feature b is found in the document. As step 33, a variable L is set to zero. At step 34A, the next feature is read in, and for this feature a calculation of 2 to the power of minus the error factor is made at step 34B. At step 34C the logarithm of the probability of the feature f occurring in class c is calculated. The results of steps 34B and 34C are multiplied together and added to L at step 34D. Step 35 causes the steps 34A to 34D to be repeated for each feature in the document. The part of equation (5) in square brackets is then calculated at step 36. If all classes have not been checked, this is determined at step 37, and a new class is then set at step 38, following which the method is repeated

for the new class. Otherwise, the class having the largest value of M is determined at step 39 to be the class to which the set of data is best classified.

The pseudocode given above for the programs relates to a simple example. Modern programming languages having data structures allowing arrays that automatically grow could be used to advantage. The invention is not limited to programs having structure like that of the above pseudocode programs.

It will be appreciated that the training program is fairly slow to run, but that the recognition program is able to process the documents much more quickly.

Although the example given calculates error factors for pairs of features, and uses these error factors in classifying documents, the invention is not so limited. For example, the training program may be arranged to calculate error factors for three features occurring together in documents belonging to a category, and/or to calculate error factors for word strings of any length.

A textual 'feature' may be a word, a string of more than one word, or it may be a member of predefined group. Such a group may embrace extensions of a word-stem, such as classif+ would embrace 'classify', 'classification', 'classified' etc. or variations of a verb, such as 'is', 'am', 'are', 'was' and 'were', all being related to the verb 'to be'.

Common features which are not expected to be relevant to classification may be ignored. In a textual document, such features may be linking or article words such as "the", "a", "if", etc.

Also, the invention is applicable to features other than textual features, and the skilled person will understand how to apply the invention to features in sets of data other than text documents.

Preferably, the categorisation system described above is included as one component of a multicomponent system, as illustrated in Figure 4. Referring to Figure 4, a

classification system 40 is shown comprising first, second and third document classification sub-systems 41 to 43. One of the systems 41 to 43 is the system of Figure 1, and the other sub-systems are alternative and different document classification systems. Each subsystem 41 to 43 is connected to receive training documents 44 which are pre-classified, and to receive unclassified documents 45. Each sub-system 41 to 43 is arranged to classify the documents 45, and to provide a suggested class to a decision module 46. A decision as to which of the suggested classes is used is then made according to a voting or democratic decision-making process running in the decision module 46. Documents 47 labelled with classifications so decided are then provided as an output of the system 40.

Claims

1. A method of classifying a set of data, the method comprising:
in a training phase:
estimating from pre-classified sets of training data the probability of at least two features occurring together in each of at least two categories; and
calculating, from the appropriate estimated probability and a measure of the probability of the features occurring together in the sets of training data if the features were independent, an error factor for each category; and
in a recognition phase: using the error factor to classify the set of data.

2. A method as claimed in claim 1, in which the calculating step comprises comparing a vector of the estimated probabilities of the at least two features occurring together in each of the categories with a vector of the probabilities, if the features were independent, of the two features occurring together in each category.

3. A method as claimed in claim 2, in which the comparing step comprises dividing the dot product of the two vectors by the product of the moduli of the two vectors.

4. A method as claimed in any preceding claim, in which the using step includes finding the product, or like function, of a function of the error factor and a measure of the estimated probability of the two features occurring together.

5. A method as claimed in claim 4, in which the function of the error factor includes the step of finding a power of a value, the power including a measure of the error factor.

6. A method as claimed in claim 4 or claim 5, in which the using step includes summing in respect of at least two feature combinations in a set of data the products, or like function, of the function of the error factor for that feature combination with the measure of the estimated probability for that feature combination.

7. A method as claimed in any preceding claim, in which the using step includes summing a measure of the probability of a set of data falling into the category being tested.

8. A method as claimed in any preceding claim in which the using step includes finding the category for which a largest value is calculated.

9. An optical, electrical, electromagnetic or magnetic signal representing machine instructions for controlling computer apparatus to perform a method according to any of the preceding claims.

10. A signal according to claim 9, comprising a temporally varying optical, electrical or electromagnetic signal.

11. A signal according to claim 9, comprising a spatial magnetic field distribution.

12. A signal according to claim 9, comprising a spatial optical characteristic distribution.

13. Apparatus for classifying a set of data, the apparatus comprising:

an estimator arranged in a training phase for estimating from pre-classified sets of training data the probability of at least two features occurring together in each of at least two categories;

a calculator arranged in the training phase for calculating, from the appropriate estimated probability and a measure of the probability of the features occurring together in the sets of training data if the features were independent, an error factor for each category; and

a classifier arranged, in a recognition phase, to use the error factor to classify the set of data.

14. Apparatus as claimed in claim 13, in which the calculator includes a comparator for comparing vector of the estimated probabilities of the at least two features occurring

together in each of the categories with a vector of the probabilities, if the features were independent, of the two features occurring together in each category.

15. Apparatus as claimed in claim 14, in which the comparator comprises means for dividing the dot product of the two vectors by the product of the moduli of the two vectors.

16. Apparatus as claimed in any of claims 13 to15, in which the classifier is arranged to find the product, or like function, of a function of the error factor and a measure of the estimated probability of the two features occurring together, and to use the result to classify the set of data.

17 Apparatus as claimed in claim 16, in which the function of the error factor involves a power of a value, the power including a measure of the error factor.

18. Apparatus as claimed in claim 16 or claim 17, in which the classifier is arranged to sum in respect of at least two feature combinations in a set of data the products, or like function, of the function of the error factor for that feature combination with the measure of the estimated probability for that feature combination, and to use the results to classify the set of data.

19. Apparatus as claimed in any of claims 14 to 18, in which the classifier is arranged to sum a measure of the probability of a set of data falling into the category being tested, and to use the result to classify the set of data.

20. Apparatus as claimed in any of claims 14 to 19, in which the classifier is arranged to find the category for which a largest value is calculated, thereby to classify the set of data.

21. A method of classifying sets of data substantially as described with reference to the accompanying drawings.

22. Apparatus for classifying sets of data substantially as shown in and/or as described with reference to Figure 1, or as modified by Figure 4, of the accompanying drawings.

28

Application No: GB 0226145.1 **Examiner:** R. F. King
Claims searched: **Date of search:** 24 April 2003

Patents Act 1977 : Search Report under Section 17

Documents considered to be relevant:

Category	Relevant to claims	Identity of document and passage or figure of particular relevance	
A	1, 13	WO00/65427 A2	[Microsoft Corp] See abstract
A	"	US2002/0091676 A1	[IBM] See abstract
A	"	US 6,388,592 B1	[IBM] See abstract
A	"	US 6,182,058 B1	[Silicon Graphics] See abstract

Categories:

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application

Field of Search:

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC[V]:

G4A

Worldwide search of patent documents classified in the following areas of the IPC[7]:

H04N

The following online and other databases have been used in the preparation of this search report:

Epoque, internet.