Automated Text Classification in the DMOZ Hierarchy

Lachlan Henderson

November 6, 2009

Abstract

The growth in the availability of on-line digital text documents has prompted considerable interest in Information Retrieval and Text Classification. Automation of the management of this wealth of textual data is becoming an increasingly important endeavor as the rate of new material continues to grow at its substantial rate. The open directory project (ODP) also known as DMOZ is an on-line service which provides a searchable and browsable hierarchically organised directory to facilitate access to the Internets' resources. This resource is considerably useful for the construction of intelligent systems for on-line content management.

In this report the utility of the publicly available Open Directory Project data for the classification of World Wide Web (WWW) text documents is investigated. The resource is sampled and a range of algorithms are applied to the task namely, Support Vector Machines (SVM), Multi-class Rocchio (Centroid), k-Nearest Neighbour, and Naïve Bayes (NB). The theoretical and implementation details of the four text classification systems are discussed. Results from the tuning and performance of these algorithms are analysed and compared with published results. Related work from the areas of both text classification and classification in general is surveyed. Some of the unique issues of large scale multi-class text classification are identified and analysed.

1 Introduction

The digitization of text information has equipped corporations, governments and individuals the ability to publish, store and share information in new ways previously not possible. Particularly the World Wide Web (WWW) is an enormous source of information, additionally large document repositories exist in patent offices, government departments, and within large corporate entities. Despite this growth in computerised information, methods for accessing and organising it have been lagging. Automated information retrieval and particularly text classification provide a means of addressing this task.

Text classification is a supervised learning technique where a model is constructed from labeled example documents. This model can then be applied to new, previously unseen documents to determine their most appropriate label. Supervised learning is an active area of research where many new and significant innovations are being made every year. Currently, there is already a broad range of techniques available many of which can be applied to the text classification problem.

Open Directory Project (ODP) is a collaborative effort of over 56,000 volunteers who contribute and renew links to WWW content for a growing list of over 760 thousand categories. This represents a considerable convenience for users of the Internet and also a valuable resource for Data Mining applications. The directories hierarchical structure provides a simple and effective way for information to be organised and accessed.

The Open Directory Project was first made available in 1998 when Richard Skrenta and Bob Truel published on-line a hand crafted directory based on Usenet news groups. Originally the directory was named GnuHoo, this name was soon abandoned due to complaints from Richard Stallman the GNU Software Foundation originator and the web search company Yahoo!. The directory was purchased by Netscape in 1998, over time though various corporate shuffles the directory is now in the ownership of Time-Warner. The ODP remains a not for profit organisation where content is overseen by a considerable community of volunteer editors. The ODP also goes under the name of DMOZ (or Dmoz) [DMO], an acronym for Directory Mozilla. Data from the ODP constitutes the basis of many other on-line directories including those provided by Google, Lycos, HotBot.

The size and number of topics in the ODP, and document databases like it provide a challenging task for researchers in finding scalable systems which can fully take advantage of their potential. Different types of hierarchical classification schemes have been applied to the ODP data and similar directories with differing levels of success [CM07, DC00, YZK03, GM05]. Successful large scale systems on hierarchical document repositories have been constructed, the United States Patent office uses one such system [Lar99] which operates on over 5 million patents consisting of 100-200 gigabytes of text.

2 The Data Set

The complete content of the DMOZ on-line directory is available as a compressed Resource Description Framework (RDF) file. Where each WWW resource provided in the on-line directory is available as a XML entry with the Uniform Resource Locator's (URL) title, description, and topic. The data is provided in Unicode with the UTF8 encoding to cater for its multilingual content of over 75 languages.

Hierarchy Level	Topic Count	Document Count	Topic (non-empty) Count
0	1	0	0
1	17	88	3
2	656	6421	335
3	7764	128888	5974
4	39946	472830	33364
5	89934	778173	77495
6	109847	785585	89268
7	167528	737462	128985
8	165460	663991	125173
9	107407	520592	86859
10	56265	332234	50205
11	15903	147529	15097
12	3906	34436	3750
13	648	5674	537
Total	765282	4613903	617045

Table 1: Frequency of Topics and Documents at Hierarchy Level

The directory organises approximately 4,613,903 URLs over 765,282 hierarchical categories. Table 1 shows the document counts at each level in the directory hierarchy. The hierarchy commences at level 0, the root called Top, at each successive level the topic is appended with an extra label separated by a / symbol. Labels at the second level are of the form Top/Shopping/Music, Top/Society/Holidays, Top/Shopping/Holidays. Document occurrence count peaks around the middle of the hierarchy at level 6, topic count peaks at level 7, beyond these points in the directory the counts decrease consistently. Not all of the topics are populated with documents, the 'Topic (non-empty) Count' column in table 1 shows the number of topics actually with documents at a particular level.

3 Algorithms

Classifier Types Text classification is the assignment of a Boolean value to the tuple $< d_j, c_i > \in D \times C$ where D is the domain of documents and $C = \{c_1, c_2, \ldots c_{|C|}\}$ a set of predefined categories. This is a task of approximating a target function $\hat{\Phi}$: $D \times C \rightarrow \{T, F\}$, which corresponds as close as possible the real function which assigns the tuple $< d_j, c_i >$ the correct Boolean value. This target function is called

a classifier and the difference between the approximated and actual target functions is termed the effectiveness of the classifier.

In the case of Supervised Learning the target function is modeled from labeled document examples. Documents must be preprocessed to transform them into a form which simplifies the computational task in a stage called feature extraction. Semi-supervised and unsupervised learning techniques which discover patterns within partially and totally unlabeled data respectively, this topic is not discussed here. Parametric methods utilise a model to describe the way the data is distributed, the task of learning is the determination of the parameters to the model. In non-parametric approaches data distributions are measured directly from the training data, and no functional form is assumed.

Classifier Desirable Properties The properties a text classification algorithm should have are computational efficiency, robustness, and statistical stability. Computational efficiency ensures that the algorithm is able to scale to larger real-world problems. Robustness refers to the requirement that the algorithm must be tolerant to a level of noise in the input data. Statistical stability is the requirement that any pattern detected by the algorithm should not be due to random associations in the data That is, the algorithm should generalise well and when applied to a new training set from the same source, the same pattern should be detected. Algorithms that do not generalise well either over-fit, or under-fit. Over-fitting occurs when the algorithm produces a model which describes the training well but test data poorly. Under-fitting is encountered when the model applied is too simple and cannot detect distinctly the patterns in the subsequent test samples.

Text classification is a well studied topic, to which many machine learning techniques have been applied. There is no algorithm that will universally provide the best performance for all classification problems. This is best encapsulated in what is termed the bias/variance trade off. Bias is due to the assumptions in the domain knowledge built into the algorithm. A large bias will produce consistently wrong classifiers if the model cannot describe the complexity of the decision surfaces within the data. Low bias may allow over fitting to occur, the classifier will work on one testing set but not another. Low variance is manifest in learning methods that produce similar decision surfaces with different training data sets. The existence of high variance on the other hand is manifest by considerable changes in the decision surfaces with different training data from the same source.

Document and Feature Representation Document representation is a central concern in classification. Typically texts cannot be used directly by classification algorithms. Documents are transformed by a procedure to represent the semantic content in a compact way optimised for numeric processing. Elements not holding any semantic relevance are typically removed from the representation. There are two main forms of representation in text classification, the vector space (VSM) and the term occurrence vector [MN98] (binary or multinomial). In the vector space model documents are vectors with one real valued element for each selected term in the document. Commonly the elements of the vector are weighted, and normalised using the TF-IDF

[SB87, HKP06] weighting scheme which in addition to weighting terms frequent in the document, also penalises terms frequent across many documents. The resulting vectors all extend as points on a unit hypersphere, the cosine of the angle between the document vectors is the typical measure of judging their similarity or dissimilarity. In the term occurrence vector each element refers either in the binary case the existence or non-existence of a term in the document, in the multinomial case the value represents the number of times the term is seen in the document. The later is often called a bag of words representation. In some cases authors [STC04] make no distinction between the two representations. It is fair to say that they are both vectorial models however the term VSM has been applied predominately to the representation where term weights are a heuristic measure of term importance that distributes points of differing classes at differing cosine angles and not a probabilistic measure of term distribution.

3.1 Centroid Based Classification

Centroid text classification is a variation of the Rocchio relevance feedback method adapted for multi-class classification [Roc71, Joa97]. It utilises the vector space model to represent documents in a $\mathbb{R}^n$ metric space. Each class is represented by a mean or centroid vector. The approach taken here is that of Han and Karypis (2000) [HK00] who have reported good accuracy in their implementation over a broad range of data sets. Documents are firstly represented by the TF-IDF scheme shown in equation 1, the term frequency here is the simple raw counts of the terms in the document. The vector is then normalized by dividing each term by the euclidean vector length $||d_{td-idf}||_2$.

The decision boundary between classes is a hyperplane, and so is a linear classifier. This type of classification assumes that the class regions are spherical with similar radii. Distributions that are multimodal do not perform well in centroid or Rocchio classification. The centroid approach can also be extended through the use of kernel functions and so benefit from all the advantages of using kernels [STC04].

$$d_{tf-idf} = (tf_1 log(n/df_1), tf_2 log(n/df_2), \ldots, tf_n log(n/df_n)) \tag{1}$$

$$cos(d_i, d_j) = \frac{d_i \cdot d_j}{||d_i||_2 * ||d_j||_2} \tag{2}$$

The cosine similarity is given by equation 2 this maps the similarity of documents with a real value ranging between 0 and 1. As the document vectors are already normalised the cosine function in equation 2 when applied to the vector here becomes the dot product $cos(d_i, d_j) = d_i \cdot d_j$.

Each centroid is the mean vector located in the middle of the cloud of vectors representing the documents for a class.

$$C = \frac{1}{||S||} \sum d \tag{3}$$

The cosine similarity function for a test document d with a centroid C for a particular class is expressed in equation 4. As the document vectors are already normalised the cosine measure can be further simplified.

$$cos(d, C) = \frac{d \cdot C}{||d||_2 * ||C||_2} = \frac{d \cdot C}{||C||_2} \tag{4}$$

The decision rule for classifying a document then becomes equation 5.

$$\underset{j=1,\ldots,k}{\operatorname{argmax}}(cos(x, C_j)) \tag{5}$$

The algorithm for the centroid classification is shown below. Simply the input data is sorted by label, then for each class calculate the mean of the TF-IDF weighted vectors. This approach is roughly linear in the input data.

Algorithm 1 centroid-train

Input: set of vectors representing the problem
Output: classifier

sort samples by label y
$ncentroid \leftarrow 0$
for $s \in samples$ **do**
  if $s.y \neq yprev \vee ncentroid = 0$ **then**
    if $ncentroid \neq 0$ **then**
      $centroid[ncentroid].\boldsymbol{x} \leftarrow centroid[ncentroid].\boldsymbol{x}/n$
      $centroid[ncentroid].y \leftarrow yprev$
      $ncentroid \leftarrow ncentroid + 1$
    end if
    $centroid[ncentroid].n \leftarrow 1$
    $yprev \leftarrow s.y$
  else
    $\boldsymbol{z} \leftarrow$ TF-IDF(s.$\boldsymbol{x}$)
    $centroid[ncentroid].\boldsymbol{x} \leftarrow centroid[ncentroid].\boldsymbol{x} + \boldsymbol{z}$
    $centroid[ncentroid].n \leftarrow centroid[ncentroid].n + 1$
  end if
end for
$centroid[ncentroid].\boldsymbol{x} \leftarrow centroid[ncentroid].\boldsymbol{x}/n$
$centroid[ncentroid].y \leftarrow yprev$
$ncentroid \leftarrow ncentroid + 1$
return classifier $centroid$

To classify an example iterate through each centroid and measure the distance with the test vector, choose the class with the smallest distance.

3.2 k-Nearest Neighbour

The k-Nearest Neighbour (k-NN) algorithm is one of the simplest algorithms to implement and always represents a good datum for comparison with other more sophisticated algorithms. It is an example of instance based learning, on-line or lazy learning. There are a number of different approaches, the one used here is the majority vote method

Algorithm 2 centroid-predict

Input: model $centroid$, vector $\boldsymbol{x}$ unlabelled
Output: label from closest centroid

1: $\boldsymbol{x} \leftarrow$ TF-IDF($\boldsymbol{x}$)
2: $bestprev \leftarrow 0$
3: $bestcentroid \leftarrow 0$
4: **for** $\boldsymbol{c} \in centroid$ **do**
5: $\quad curr \leftarrow \boldsymbol{c} \cdot \boldsymbol{x}$
6: $\quad$ **if** $curr > bestprev$ **then**
7: $\quad\quad bestprev \leftarrow curr$
8: $\quad$ **end if**
9: **end for**
10: **return** $centroid[bestcentroid]$

where the most frequent class is chosen from K nearest neighbours. Other approaches include weighting each nearest neighbour inverse proportionally by the distance to the test point and weighting the nearest neighbours by their ability to classify. The k-NN algorithm is bounded by twice the Bayes error, and asymptotically approaches the Bayes error.

The algorithm is linear with the number of example vectors or exemplars used, assuming that the distance measure is inexpensive to calculate. This is often not the case for problems expressed in high dimensional spaces. There are a number of variations to improve the run time of prediction they roughly form tttwo groups, either removing exemplars which do not contribute to describing the decision surfaces in the data or faster means of finding the nearest neighbours by performing a metric search [Yia93]. The k-NN approach is a localised approximation of target function, no attempt is made to form a target function over the entire instance space, this has advantages when the the target function is extremely complex.

Algorithm 3 knn-train

Input: samples
Output: classifier $exemplar$

1: $nexemplar \leftarrow 1$
2: **for** $s \in samples$ **do**
3: $\quad exemplar[nexemplar].\boldsymbol{x} \leftarrow s.\boldsymbol{x}$
4: $\quad exemplar[nexemplar].label \leftarrow s.label)$
5: $\quad nexemplar \leftarrow nexemplar + 1$
6: **end for**
7: **return** classifier $exemplar$

Algorithm 4 knn-predict

Input: Vector $\boldsymbol{x}$ to be classified, parameter k
Output: Most frequent label of k-Nearest Neighbours

for $i \in [1 \ldots nexemplar]$ **do**
  $exemplar[i].similarity \leftarrow similarity(exemplar[i].\boldsymbol{vector}, \boldsymbol{x})$
  $exemplar[i].count \leftarrow 0$
end for
sort $exemplar$ length $nexemplar$ by similarity descending
sort $exemplar$ length k by label descending
$lastlabel \leftarrow none$
for $i \in [1 \ldots k]$ **do**
  if $exemplar[i].label \neq lastlabel$ **then**
    $j \leftarrow i$
    $exemplar[j].count \leftarrow= 1$
  else
    $exemplar[j].count \leftarrow exemplar[j].count + 1$
  end if
end for
sort $exemplar$ length k by count descending
return $exempar[1].label$

3.3 Naïve Bayes

Naïve Bayes (NB) classifiers provide a probabilistic model with an explicit statement of simplifying assumptions. NB has two common types of document representation, the Bernoulli and multinomial. The Bernoulli approach is to represent documents by the binary occurrence of features, the multinomial approach utilises feature counts. It has been shown that typically the multinomial approach out-performs the Bernoulli one [MN98]. Both approaches assume that documents can be modelled as probability distributions of independent term events. The task is then to estimate the parameter of these distributions by observing there occurrence in the training data. The Naïve assumption is that each word event is independent of its context within the document. This assumption although contrary to intuition does not typically affect the effectiveness of the algorithm significantly. Generally it has been observed that word n-grams although have good semantic properties have poor statistical properties.

Both the train and predict run times are proportional to the term space and not with training data set size. Bayes theorem can be written as shown in equation 6

$$P(Y = y_i|X = x_k) = \frac{P(X = x_k|Y = y_i)P(Y = y_i)}{P(X = x_k)} \tag{6}$$

According to the theory of total probability the $P(X = x_k)$ term can be expressed as shown in equation 7.

$$P(Y = y_i|X = x_k) = \frac{P(X = x_k|Y = y_i)P(Y = y_i)}{\sum_j P(X = x_k|Y = y_j)P(Y = y_i)} \tag{7}$$

In the case of text classification we are interested in estimating the probabilities over terms $P(X_1 \ldots X_n|Y = y_i)$ which represent the conditional probability of a set of terms X occurring given a class Y. The NB assumption is that the probability of $P(X_1 \ldots X_n|Y = y_i)$ can be approximated by the expression $\prod_i P(X_i|Y = y_i)$. The assumption is that the occurrence of a term is conditionally independent of all the other terms.

$$P(Y = y_k|X1 \ldots X_n) = \frac{P(Y = y_k) \prod_i P(X_i|Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i|Y = y_j)} \tag{8}$$

The task in generating a naïve Bayes model is to estimate the probabilities by counting the terms for documents of a category, the set of parameters are of the form shown in equations 9 and 10 where i,j and k are indexes for terms, documents, and labels respectively.

$$\hat{\theta}_{ijk} = P(X_i = x_{ij}|Y = y_k) = \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\}}{\#D\{Y = y_k\}} \tag{9}$$

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\}}{|D|} \tag{10}$$

The decision rule is:

$$Y \leftarrow \underset{y_k}{\text{argmax}} P(Y = y_k) \prod_i P(X_i|Y = y_k) \tag{11}$$

Algorithm 5 nbayes-train

Input: Labeled samples as sparse vectors with elements $< id, value >$ where $id \in [1 \ldots nattr]$

Output: Classifier $model$

```
for s ∈ samples do
    for e ∈ s do
        count[e.id][s.label] ← count[e.id][s.label] + e.value
        count[nattr + 1][s.label] ← count[nattr + 1][s.label] + e.value
        D ← D + e.value
        model.J ← nattr
        model.K ← ncat
        model.D ← D
        model.count ← count
    end for
end for
return model
```

Algorithm 6 nbayes-predict

Input: model m, parameters p, and vector $\boldsymbol{x}$ to be classified
Output: best category

for $c \in [1 \ldots m.ncat]$ **do**
  $score[c] \leftarrow log(m.count[m.nattr+1][c] + p.l) - log(m.D + p.l * m.k)$
end for
for $e \in \boldsymbol{x}$ **do**
  for $c \in ncat$ **do**
    $score[c] \leftarrow score[c] + log(m.count[e.id][c] + p.l)$
    $score[c] \leftarrow score[c] - log(m.count[m.nattr+1][c] + p.l * m.J)$
  end for
end for
$best \leftarrow 0$
for $c \in [1 \ldots m.ncat]$ **do**
  if $score[best] < score[c]$ **then**
    $best \leftarrow c$
  end if
end for
return $best$

3.4 Support Vector Machines

The technique of support vector machines (SVM) was first introduced by Vapnik. It is based on the computational learning theory principle of structural risk minimization. In the operational setting this is the solving of a quadratic programming problem (QP) to find what is termed a soft margin separating two classes. The soft margin refers to a maximal margin that envelopes the hyperplane defined by key vectors called support vectors. Allowance is made for the non linearly separable case by penalising examples lying on the wrong side of the hyperplane. Several approaches to generalising this method to the multi-class setting fall generally into two types: one against one or one against many. Generally the one against one approach is used, of which there is also two types the directed acyclic graph approach [PCSt00] and a voting scheme where each of $n(n-1)/2$ outcomes is a vote for a particular class.

Given a set of examples $(X_i, y), i = 1, \ldots, l$ where $v_i \in \mathbb{R}^n any \in \{1, -1\}^l$ a maximal margin hyperplane separating the data into different regions representing the different classes is defined as the following optimization problem shown in equation 12.

$$\begin{gathered} \min_{w,b,\xi} \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ subject\,to\, y_i(\mathbf{w}\phi(x_i) + b) \leq 1 - \xi_i \\ \xi_i \leq 0 \end{gathered} \tag{12}$$

The dual is.

$$\min_{\alpha} \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} \boldsymbol{w} - e^T \alpha$$

$$subject\ to\ y^T \alpha = 0 \tag{13}$$

$$0 \leq \alpha_i \leq C, i = 1, \ldots l$$

Where e is a vector of all ones, C is the upper bound, Q is an l by l positive semi-definite matrix, $Q_i j \equiv y_i y_j$ and $K(\boldsymbol{x_i}, \boldsymbol{x_j}) \equiv \Phi(\boldsymbol{x_i})^T \Phi(\boldsymbol{x_j})$ is the kernel.

The decision function is,

$$sgn\left(\sum_{i=1}^{l} y_i \alpha_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b\right) \tag{14}$$

The SVM implementation used here is LibSVM [CL01] which can perform multi-class classification using a fast QP approach called sequential minimal optimization (SMO).

3.5 Kernelised k-Nearest Neighbour and p-Spectrum Kernel

The simplest form of the k-NN algorithm uses the euclidean distance equation 15 for ranking documents according to similarity.

$$d(\boldsymbol{x_1}, \boldsymbol{x_2}) = ||\boldsymbol{x_1} - \boldsymbol{x_2}||_2 = \sqrt{\boldsymbol{x_1} \cdot \boldsymbol{x_1} - 2 * \boldsymbol{x_1} \cdot \boldsymbol{x_2} + \boldsymbol{x_2} \cdot \boldsymbol{x_2}} \tag{15}$$

The dot products in this equation can be substituted with kernel functions via the kernel trick [SS01] as shown in the equation 16

$$d(\boldsymbol{x_1}, \boldsymbol{x_2}) = \sqrt{K(\boldsymbol{x_1}, \boldsymbol{x_1}) - 2 * K(\boldsymbol{x_1}, \boldsymbol{x_2}) + K(\boldsymbol{x_2}, \boldsymbol{x_2})} \tag{16}$$

If $\boldsymbol{x_1}$ represents the test string, the first dot product is constant for the test string, the last dot product term is constant for each exemplar in the k-NN model. The first and last terms can be stored and reused at differing times as an optimisation.

In the context of document classification the bias of the document lengths for $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ can be reduced by normalising the kernel functions as in equation 17 to give equations of the form 18.

$$\hat{K}(x_1, x_2) = \frac{< \Phi(x_1), \Phi(x_2) >}{||\Phi(x_1)|| * ||\Phi(x_2)||} \tag{17}$$

$$\hat{K}(x_1, x_2) = \frac{K(\boldsymbol{x_1}, \boldsymbol{x_2})}{\sqrt{K(\boldsymbol{x_1}, \boldsymbol{x_1}) * K(\boldsymbol{x_2}, \boldsymbol{x_2})}} \tag{18}$$

The use of character n-grams in formulations to compute document similarity has been studied in a number of publications and have shown to offer a way to language independent text classification [KSSI05, OVS03, TV06, STC04]. Essentially this is encapsulated as a kernel function of the form shown in 19

$$\hat{K}(x,x') = \sum_{s \sqsubseteq x, s' \sqsubseteq x'} s_s \delta_{s,s'} = \sum_{s \in A^*} w_s num_s(x) num_s(x') \tag{19}$$

That is, counting the number of occurrences of every substring in both x and x' weighted by w It has been observed that in document classification it is sufficient to only compare substrings to a smaller length p, this is the p-spectrum kernel. The use of suffix arrays to improve computational performance provide a simple to implement equivalent means to perform the required substring matching. An algorithm for the required processing is given in algorithms 7 and 8.

Algorithm 7 suffix-array-build

Input: string, n the length of the string
Output: array of suffixes sorted alphabetically

1: **for** $i \in [1 \ldots n]$ **do**
2: $s \leftarrow$ substring at $[i \ldots n]$
3: $l \leftarrow$ substring length $n - (i-1)$
4: $tmp[i] \leftarrow < s, l, i >$ {parameters required by sort}
5: **end for**
6: **sort** tmp alphabetically using s and l
7: **for** $i \in [1 \ldots n]$ **do**
8: $array[i] \leftarrow$ index of string given by last member of tuple $tmp[i]$
9: **end for**
10: **return** $array$

4 Empirical Evaluation

Four algorithms were used in this study, three of which were implemented by the author for this project, k-NN, centroid, and NB. The other algorithm, SVM was implemented by [CL01] and available as a library called LibSVM. LibSVM was interfaced to the same general frame work as the other implementations to allow a homogeneous treatment of the four. Only the linear kernel for the SVM was trialed.

File	knn		centroid		nbayes		libsvm	
	Det.	Ref.	Det.	Ref.	Det.	Ref.	Det.	Ref.
oh0.wc.arff	87.1±0.2	84.4	90.7±0.2	89.3	89.2±0.2	89.1	89.7±0.1	-
oh5.wc.arff	84.1±0.1	85.6	87.0±0.2	88.2	84.5±0.2	87.1	89.9±0.3	-
oh10.wc.arff	76.8±0.3	77.5	81.0±0.3	85.3	82.0±0.2	81.2	81.5±0.2	-
oh15.wc.arff	79.3±0.3	81.7	82.8±0.3	87.4	81.6±0.3	84.0	83.9±0.1	-

Table 2: Classifier 10-fold Cross Validation % Accuracy on the OHSUMED Collection Compared With Experimental Results of Karypis et. el. (95 % Confidence Limits)

Table 2 shows a comparison of the determined accuracy of the implemented algorithms with the referenced results reported by Han and Karypis (2000) [HK00] on

Algorithm 8 suffix-array-search

Input: string1, n1, string2, n2, sarray (of string2)
Output: count, the number of occurrences of string1 in string2

```
1:  i ← 1
2:  min ← 0
3:  max ← n2
4:  count ← 0
5:  found ← 0
6:  repeat
7:     i ← (min + max)/2
8:     if string1[1...n1] = string2[sarray[i]...n1] then
9:        found ← 1
10:       count ← count + 1
11:       j ← i − 1
12:       while string1[1...n1] = string2[sarray[j]...n1] do
13:          count ← count + 1
14:          j ← j − 1
15:       end while
16:       j ← i + 1
17:       while string1[1...n1] = string2[sarray[j]...n1] do
18:          count ← count + 1
19:          j ← j + 1
20:       end while
21:    else if string1[1...n1] > string2[sarray[i]...n1] then
22:       i ← i + 1
23:    else
24:       i ← i − 1
25:    end if
26: until min > max ∨ found
27: return count
```

the OHSUMED dataset *. The parameters used were k=25 (KNN), l=1.1 (NB), C=1.2 (SVM). The algorithms implemented show a similar classification accuracy to those of Han and Karypis.

The ODP compressed data was downloaded † and sampled vertically down the sub-topics at a number of sample rates from level 2 of the directory. The second level of the directory hierarchy contains 656 topics table 1 shows the general distribution of the documents under each label. Training data was drawn from the descriptions provided for each URL in the compressed data. Test samples were downloaded, parsed using the XML2 parser and the UTF8 strings from the resultant title, keywords, and description tags were extracted and concatenated space separated. Test samples with fewer than 50 strings were discarded. The final results utilised testing data sampled at the rate of 1 URL in 1000, resulting in 236 test samples, and 49 labels. The training data was sampled at the rate of 1 in 50 descriptions with a total of 91,337 samples, and 469 labels. This number of samples was chosen because the limited time available for running simulations. They give a reasonable representation of the performance of algorithms trialed. In a different setting higher sample rates would have been used. All labels in the test samples were well represented in the training data set.

The distribution of the URLs and their descriptions of the 656 topics at level 2 of the directory are shown in Figure 1. The topic names are replaced with a sequential number due to limited space on the plot x-axis. This shows a very irregular distribution of URLs over the range of topics. Table 3 shows the top ten most frequent categories which account for approximately 56% of the total document count in all 656 level 2 topics ‡. The representation of languages other than English shows that the directory is truly multilingual with Japanese included with some of the more frequently spoken European languages.

Level 2 Label	Document Count
Top/Regional/North_America	694831
Top/World/Deutsch	501512
Top/Regional/Europe	285859
Top/World/Français	233029
Top/World/Italiano	203114
Top/World/Japanese	184078
Top/World/Español	163367
Top/Society/Religion_and_Spirituality	103862
Top/World/Nederlands	97338
Top/Arts/Music	80618
Total	2547608

Table 3: Top 10 Label Document Count

As the distributions of the training and testing data sets are drawn from differ-

*The OHSUMED data set converted by George Forman, HP Labs into arff format is downloadable at http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html and is the same as that used by Han and Karypis

†Data file content.rdf.u8.gz was downloaded from the ODP web site August 2 2009

‡Top/Adult topics were not included

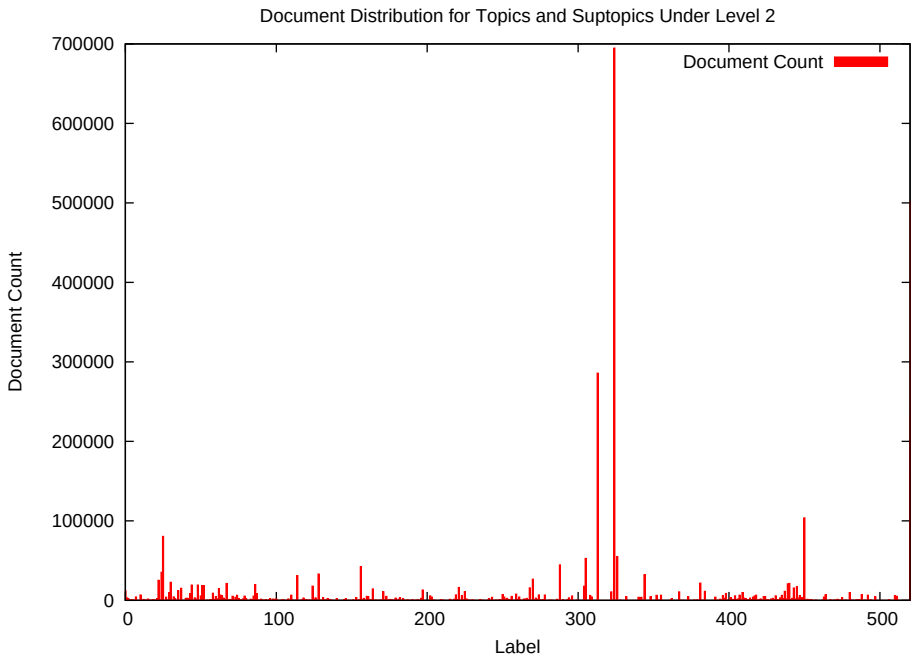

Figure 1: Document Distribution at and below Level 2 of The Hierarchy

ent sources the usual k-fold cross validation technique can't be applied, instead a re-sampling algorithm 9 is used.

The k-Nearest Neighbour parameter tuning plot Figure 2 shows initially a reasonable prediction accuracy at low training data size. The parameter plot show that k-NN is unstable at low k, indicating poor separation of the different topics at the localised area around the test sample. This may be in part due to the asymmetry of the test and train data, in effect the test sample is located badly. With increased k there is an asymptotic increase in accuracy to approximately 60% for the TF-IDF exemplars and approximately 55% for the more simpler normalised TF representation. At the 60% mark and beyond there is little improvement, perhaps at this point the capacity of the feature space is reached. Comparing this with the accuracy shown in the OHSUMED evaluation where features selection included stemming and stop word removal possibly indicates the feature space is too sparse. The best performance is with TF-IDF and approximately k=40.

Figure 3 shows the accuracy response with change in the MAP parameter l for the Dirichlet prior over observed term frequencies. The best accuracy performance is distinctly seen at a l setting of approximately 0.01. This value gives a default setting for unseen features, such a low setting shows that only little smoothing is required.

The C parameter of SVM is the intolerance given to training samples place the incorrect side of the trained decision surface. The optimum C parameter sits around 2.0 beyond this point there is no improvement.

Algorithm 9 kfold-resample

Input: $test$, $train$ data sets, $ftest$, $ftrain$, test and train sample fractions, $ntrials$
Input: train-method, predict-method
Output: mean,sd

```
for t ∈ [1 ... ntrials] do
    model ←train-method(train)
    correct ← 0
    total ← 0
    test ←random-sample(test,ftest)
    train ←random-sample(train,ftrain)
    for s ∈ test do
        label ←predict-method(s)
        if s.label = label then
            correct ← correct + 1
        end if
        total ← total + 1
    end for
    outcome[t] ← correct/total
end for
return mean-and-sd(outcome,ntrials)
```

The kernelised k-NN tuning curve shows no improvement in accuracy for the substring lengths tested. Surprisingly, no advantage is seen, as the word prefixes, suffixes and other variations in terms should be circumvented by looking into the terms via substrings, should provided a richer feature space and so higher classification effectiveness. The run-time on the k-fold cross validation on the string kernel is so long there is limited opportunity to perform a grid search over the k-NN parameter k and the maximum substring length p which may have been an important factor.

Of the four algorithms applied to the sampled data SVM and NB performed the best, SVM performed marginally better at higher training data set sizes. The TF-IDF document representation performed best across all of the algorithms, significantly better on k-NN.

The change in training time with training set size in Figure 7 shows a relatively constant NB training time. Also shown is the slow but steady growth of the centroid and k-NN times indicating only a moderate scalability with training data size. The SVM implementation shows initially a near exponential increase which then appears asymptotically level over 100 times that of the Bayesian classifier. With a larger number of topics the NB run-times and space requirements increase significantly, this is particularly important for scaling classification over larger samples of the ODP data.

The varying of prediction time with training set size is given in Figure 8. Surprisingly the SVM has a similar profile to that of k-NN indicating very poor scalability. Using a hierarchical approach, and building a classifier for each node as suggested by some authors [DC00, CM07] would be the only way to scale this method. NB oscillates around the 1 millisecond mark. This slight varying is with differing attribute

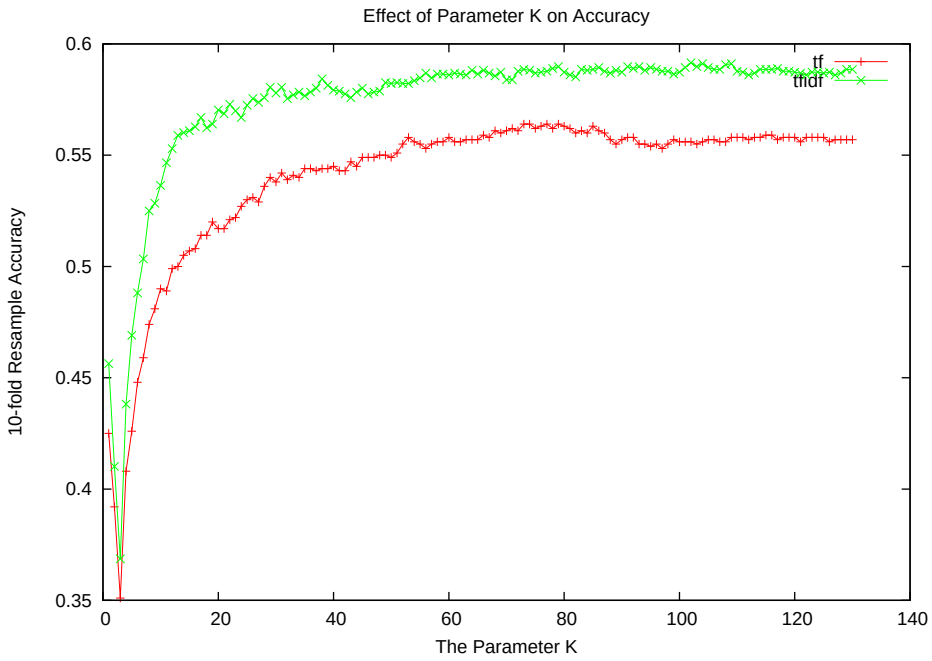

Figure 2: KNN Parameter Tuning

counts over the test and training cross validation samples. The centroids method remains steady with a slight gradient as new labels are discovered in the increasing training dataset size and is the most scalable of the algorithms.

5 Related Work

Classification Classification as a significant topic in computer science in the early years is described in Duda and Hart [DH73], where Bayesian methods are described, and early descriptions of kernel methods are also found. Many recent texts have been written [HKP06, WF05, Mit97] which give a good account of the practical task of implementing and using machine learning methods.

Text Classification Sebastiani [SR02] provides an excellent overview of the whole topic of text categorization and an excellent start to research in this area, detailing the complete process in a practical and helpful way. Information Retrieval and classification methods overlap in many areas texts in this domain [MRS08] provide helpful information on classification and document representation. Joachims [Joa98] has investigated the application of SVM to text classification in his paper 1998 he observes that in text classification all features are relevant and that dimensionality reduction in this context will only hurt performance. As SVM can handle very sparse feature vec-

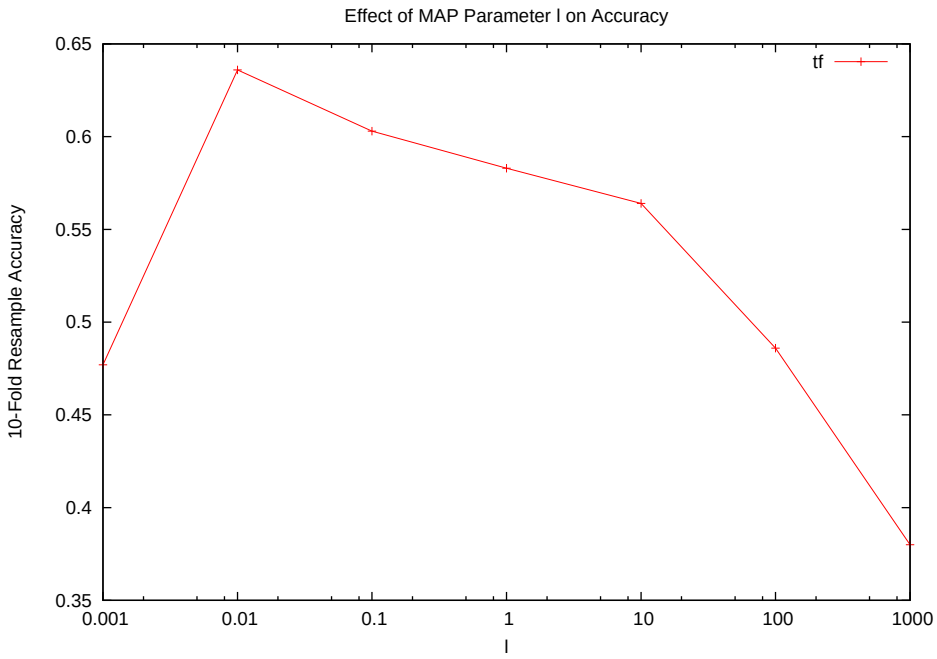

Figure 3: Naïve Bayes Parameter Tuning

tors it is a ideal algorithm to handle the text classification problem. Joachims compares NB, Rocchio, k-Nearest Neighbour and C4.5 decision tree algorithms in a binary classification setting. McCallum and Nigam [MN98] investigated the two main document representations for NB text classification the Bernoulli and multinomial. They conclude that the multinomial approach is superior in accuracy in most cases. In the case of small training data experiments the Bernoulli model performed marginally better.

Hierarchical Text Classification The problem of text classification in class hierarchies has been discussed by a small number of authors. McCallum Et. Al. [MN98] applied the statistical method of shrinkage to NB text classification. In their approach using the Yahoo directory they work from leaf nodes back up to the root in the hierarchy to smooth maximum likelihood estimates at the leaf nodes. Ceci and Malerba investigated ways to take advantage of the information encoded in the hierarchy with a greedy search algorithm which traverses the hierarchy for the best category fit for a test sample.

Regularised Text Classification Genkin, Lewis and Madigan [GAL+07] use regularized logistic regression to perform text classification. The regularising occurs by penalising complex model parameters, Genkein et. al. use a method called lasso shrinkage by using an L1 penalty term on the logistic regression parameter vector. This in

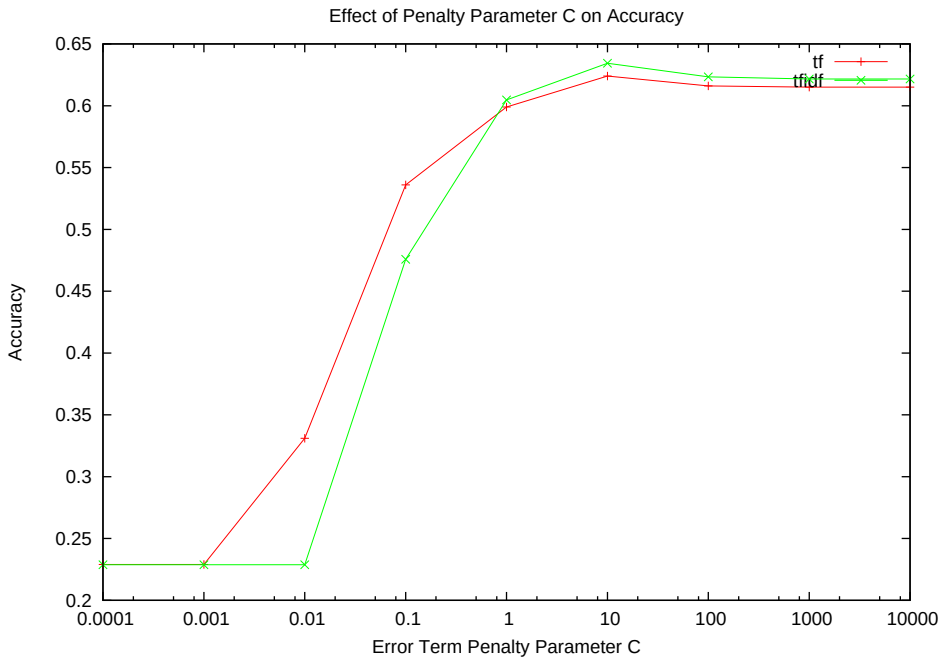

Figure 4: SVM Parameter Tuning

effect removes less important features from the logistic model, providing a sort of feature selection built into the learning process. Ifrim et. al. [IBW08] uses a branch and bound search in the space of word or character n-grams in an integrated regularised complex feature selection learning process.

String Kernels String kernels are discussed by Kruengkrai et. al. [KSSI05] in the context of language recognition. Lodhi et. al. [LSST$^+$02] investigate subsequence string kernels and compare their performance with word kernels, and n-gram kernels. Teo [TV06] investigated efficient string n-gram kernels using suffix arrays, integrating the whole into a linear time string kernel algorithm.

Nearest Neighbour Search Nearest neighbour and k-Nearest Neighbour algorithms are some of the first machine learning approaches to classification. Extensions to these algorithms through the use of kernels are offered by a number of authors [YJZ02, KSSI05, STC04]. These replace the occurrence of the dot product with a kernel to map the problem into a higher dimensional space. Making the nearest neighbor search efficient has been the topic of many papers. Relatively more recent are the vantage point methods outlined by Yianilos [Yia93] which shows the use of generalised metric spaces to enable the recursive construction of a tree data structure enabling logarithmic search performance.

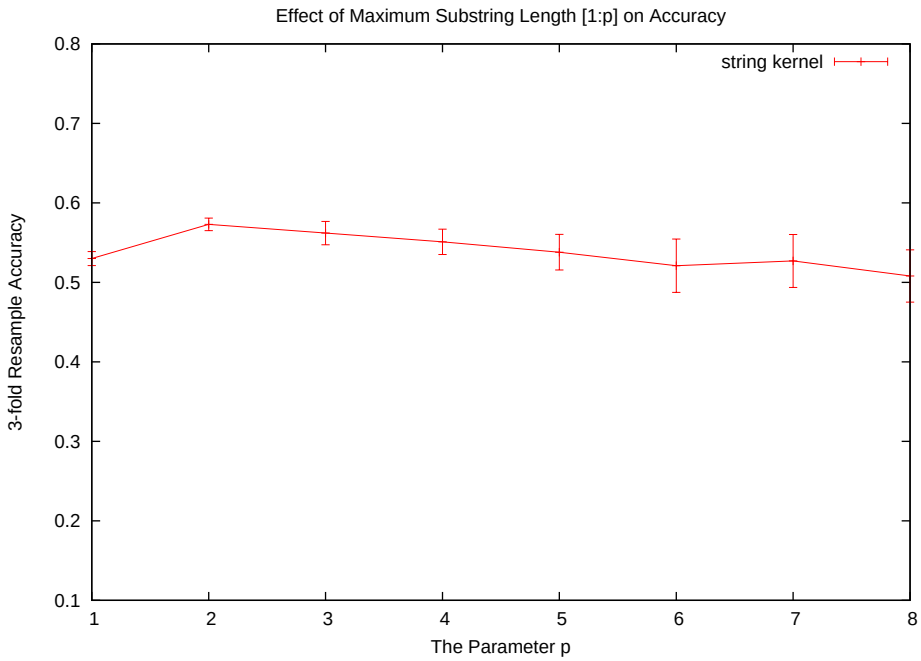

Figure 5: Kernelised k-NN with p-Spectrum String Kernel Parameter Tuning

6 Conclusions

Four algorithms were applied to the classification task of assigning web pages to topics of a small subset of the DMOZ topic hierarchy using the topic descriptions as training data. Three different schemes were feature space mappings were trialed and their performance noted. The TF-IDF approach was more effective than the TF features, this corresponds with the literature. Feature selection techniques might offer much greater improvements. However the richness of the description feature space might not be adequate to sustain higher classifier performance. The string kernel approach did not immediately offer a remedy to the limited accuracy of the classifiers over the data, however there are still many different parameters and variations left untried.

Tuning of model parameters is an essential task, which ensures optimal classifier model performance. NB, SVM, and k-NN had each one parameters to tune. SVM was utilised using a linear kernel, and so no kernel parameter tuning was required. The Centroids method as implemented had no parameters, and so was the easiest to use. The NB parameter l only required a limited range of values, where as the SVM C parameter requires a broad search over many orders. There may have been some drift in classifier performance with the parameter settings used over different training data size, this interaction was not investigated.

The effectiveness and run-time of the each algorithm with training set size was investigated. For the data sampled from the ODP the SVM and k-NN algorithms did

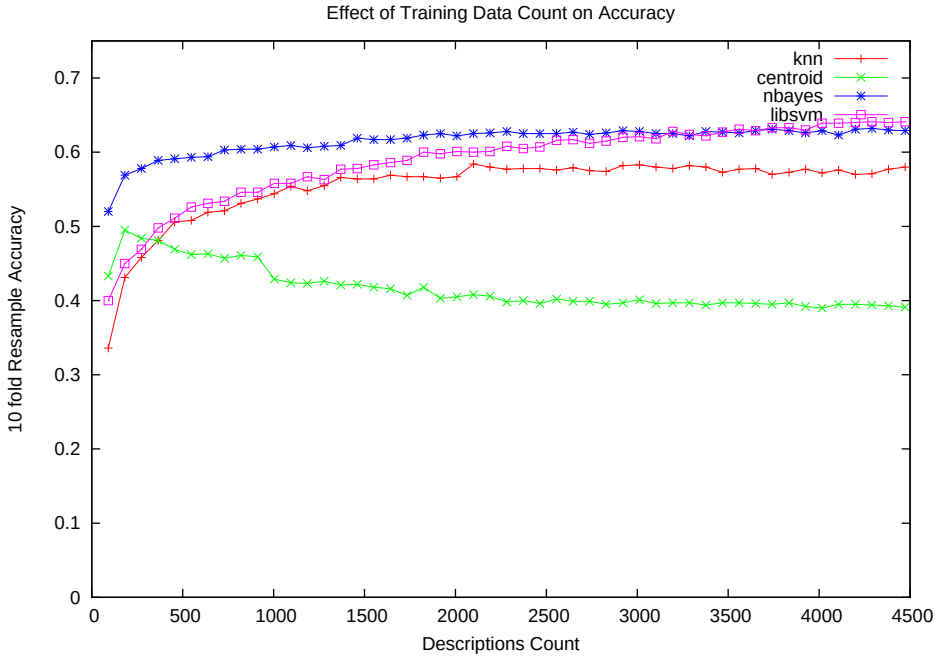

Figure 6: Effect of Training Dataset on Accuracy

not scale. The NB and the centroids approaches performed well. The run time of the kernelised k-NN with string kernel was very long, with no increase in accuracy. The advantages seen in the NB approach would soon be lost over even a small percentage of the total topic count in the ODP. The centroids approach is possibly the only one of the four would might scale, particularly with the use of a vantage method or similar algorithm.

The centroids approach appeared sensitive to the sparsity of the term space. The classification performance actually dropped with training data set size. A possible explanation for this may due to an increase of the dimensionality of the centroid vector, with each new vector in the supporting set the sparsity of the term space pushes the centroid toward the origin. The signal to noise ratio is in effect lowered and the similarity measure does not discriminate the correct topic. Another possibility is that the topics are multimodal and so with increasing training set size the different modes are more pronounced and dissipate the centroid's discriminating power.

NB, centroids and k-NN have the advantage of being incrementally adaptable with new training instances, this is not available in SVM at least in its conventional form. With additional NN searching optimisations the centroids and k-NN methods would scale well over both the instance and topic count.

The challenge of language independent classification is a critically important challenge in text classification. The utilisation of word stemming and stop word lists cannot be applied efficiently if at all to multilingual problems such as web page classification.

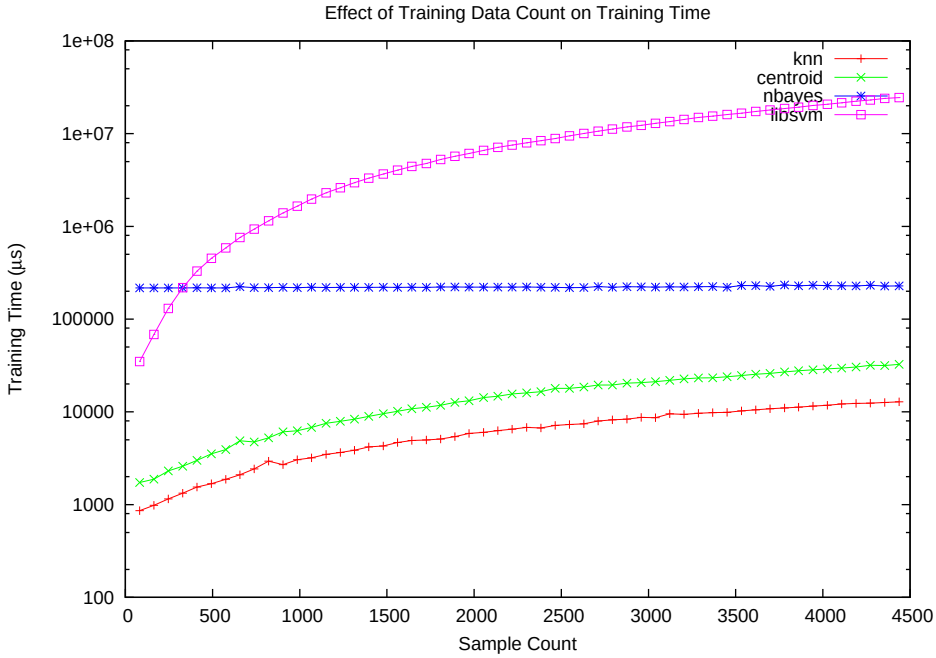

Figure 7: Train Time During Cross Validation On Training Set

Asian languages such as Japanese and Chinese are not delimited texts, agglutinative languages such as Turkish and to some extent German are not easily normalised. This important issue is often ignored in the broad text classification field.

Each classification problem has its unique challenges, there is no general solution to apply. The results of this investigation highlight the limitations of conventional classification algorithms on even a small subset of a real-world text corpus.

References

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[CM07] M. Ceci and D. Malerba. Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, 28(1):37–78, 2007.

[DC00] Susan Dumais and Hao Chen. Hierarchical classification of web content. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263, New York, NY, USA, 2000. ACM.

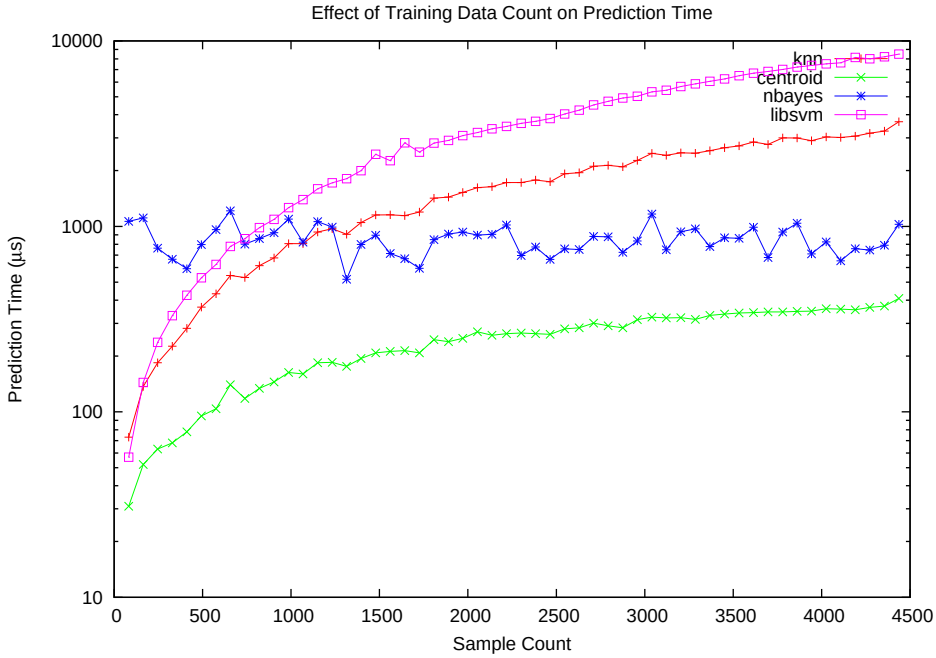

Figure 8: Predict Time During Cross Validation On Training Set

[DH73] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. A Wiley-Interscience Publication, New York: Wiley, 1973, 1973.

[DMO] The open directory project (http://www.dmoz.org/).

[GAL+07] Genkin, Alexander, Lewis, D. David, Madigan, and David. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, August 2007.

[GM05] Marko Grobelnik and Dunja Mladenic. Simple classification into large topic ontology of web documents. *CIT*, 13(4):279–285, 2005.

[HK00] Eui-Hong Han and George Karypis. Centroid-based document classification: Analysis and experimental results. In *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 424–431, London, UK, 2000. Springer-Verlag.

[HKP06] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques, Second Edition)*. Morgan Kaufmann, 2006.

[IBW08] Georgiana Ifrim, Gökhan Bakir, and Gerhard Weikum. Fast logistic regression for text categorization with variable-length n-grams. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 354–362. ACM, 2008.

[Joa97] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98*, pages 137–142. Springer Verlag, 1998.

[KSSI05] C. Kruengkrai, P. Srichaivattana, V. Sornlertlamvanich, and H. Isahara. Language identification based on string kernels. In *Communications and Information Technology, 2005. ISCIT 2005. IEEE International Symposium on*, volume 2, pages 926–929, Oct. 2005.

[Lar99] Leah S. Larkey. A patent search and classification system. In *DL '99: Proceedings of the fourth ACM conference on Digital libraries*, pages 179–187, New York, NY, USA, 1999. ACM.

[LSST+02] 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.

[Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[MN98] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification, 1998.

[MRS08] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[OVS03] Vishwanathan Dept Of, S. V. N. Vishwanathan, and Alexander J. Smola. Fast kernels for string and tree matching. In *Advances in Neural Information Processing Systems 15*, pages 569–576. MIT Press, 2003.

[PCSt00] John C. Platt, Nello Cristianini, and John Shawe-taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, pages 547–553. MIT Press, 2000.

[Roc71] J. J. Jr. Rocchio. *Relevance feedback in Information Retrieval*. Prentice-Hall Inc., 1971.

[SB87] Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA, 1987.

[SR02] Fabrizio Sebastiani and Consiglio Nazionale Delle Ricerche. Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47, 2002.

[SS01] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[STC04] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[TV06] Choon Hui Teo and S. V. N. Vishwanathan. Fast and space efficient string kernels using suffix arrays. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 929–936, New York, NY, USA, 2006. ACM.

[WF05] I.H. Witten and E. Frank. Data mining: Practical machine learning tools and techniques, 2005.

[Yia93] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 311–321, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[YJZ02] Kai Yu, Liang Ji, and Xuegong Zhang. Kernel nearest-neighbor algorithm. *Neural Process. Lett.*, 15(2):147–156, 2002.

[YZK03] Yiming Yang, Jian Zhang, and Bryan Kisiel. A scalability analysis of classifiers in text categorization. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 96–103, New York, NY, USA, 2003. ACM.