BUILDING A WEB PORTAL USING SUPPORT VECTOR MACHINES

by

Joseph Muthu

B.Tech Chemical Engineering, Anna University, 1993

A thesis submitted to the Faculty of the Graduate School of the University of Colorado at Colorado Springs in partial fulfillment of the requirements for the degree of Master of Science

Department of Computer Science 2003
This thesis for the Master Of Science degree by
Joseph Muthu
has been approved for the
Department of Computer Science
by

Advisor: Jugal Kalita

Marijke F. Augusteijn

C. Edward Chow

Date ________________
Joseph Muthu,  (MS, Computer Science)

Building a Web Portal using Support Vector Machines

Thesis directed by Professor Jugal Kalita

The world wide web is growing rapidly, manual classification of web pages is very time consuming and maintenance is high. Existing portal sites are not catering to the needs of researchers and entrepreneurs interested in Technology Transfer. Automatic classification has been widely researched and will eventually be the way to classify web content in the future. Support vector machines with its powerful learning capability have shown practical relevance for classification and regression problems. A web portal model has been built using this machine. It uses web document along with its citing document for learning and classification. Experiments were conducted on this model to evaluate its performance. The results have confirmed and quantified the practical applicability of this model. The model has been designed in such a way that it can be easily trained for classifying web documents for different technology fields.
## CONTENTS

### CHAPTER

I  INTRODUCTION ............................................. 1

II  OBJECTIVE ............................................... 2

III SUPPORT VECTOR MACHINES .......................... 3

   Linear SVM Model ......................................... 3

   Lagrangian Theory and Karush-Kuhn-Tucker theory .... 5

   Non-linear SVM ........................................... 8

   Imperfect Separation ..................................... 8

   Checking KKT conditions without using threshold b .... 11

IV  OPTIMIZATION METHODS ............................... 13

   Chunking .................................................. 13

   Decomposition Methods ................................ 14

   Sequential Minimal Optimization .................... 14

V  SEQUENTIAL MINIMAL OPTIMIZATION ................ 15

VI  MULTI-CLASS CLASSIFICATION ....................... 23

VII INPUT REPRESENTATION AND FEATURE SELECTION ... 25
Fetching web pages .......................... 58
Fetching anchor/extended text ............... 59
Training/Testing the data ................... 59
Crawling for new links ...................... 61
# TABLES

<table>
<thead>
<tr>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Number of web pages for each category</td>
</tr>
<tr>
<td>2. Word count for different Document Frequency</td>
</tr>
<tr>
<td>3. Accuracy obtained for Kernel and Information Gain parameters</td>
</tr>
<tr>
<td>4. Accuracy obtained for various parameters</td>
</tr>
<tr>
<td>5. Web Page Classification Comparison</td>
</tr>
<tr>
<td>6. Web Page Classification Comparison between extended text and web page</td>
</tr>
<tr>
<td>7. Webkb Dataset</td>
</tr>
<tr>
<td>8. Webkb Dataset Links Distribution</td>
</tr>
<tr>
<td>9. Webkb Dataset Results</td>
</tr>
<tr>
<td>10. Comparison of different methods</td>
</tr>
<tr>
<td>11. Comparison of SVM Methods</td>
</tr>
<tr>
<td>12. No. of words common between input types</td>
</tr>
<tr>
<td>13. Classification of Test Data</td>
</tr>
<tr>
<td>14. Classification Accuracy of Test Data</td>
</tr>
</tbody>
</table>
FIGURES

Figure

1  Web Portal System Overview ............................... 28
CHAPTER I

INTRODUCTION

The world wide web has highly relevant content to researchers, entrepreneurs, companies, administrators, and others interested in Technology Transfer. Currently the search engines like yahoo are very generic covering wide range of fields. A web portal model for Technology Transfer or a specialized field like bioinformatics would have high demand. Existing technology related search engines like research index do not categorize the web documents. There is a need for an automatic classification model for technology related web documents and also to keep up with the fast growing on-line document data.
CHAPTER II

OBJECTIVE

The objective of this project is to develop an automatic web classifier, compare the learning machine with other methods and implement this model for a particular technology field. The model would require the development of the following programs

1. Focused Crawler: It crawls the web. The program does not crawl the web indiscriminately. As it fetches and stores documents from the web for analysis, it gets feedback from the data converter on the webpage whether it is of any value or relevance. This way it identifies hypertext nodes that are superior access points.

2. Data Converter: This program would take the pages reported by the crawler and filter the data based on some parameters and convert the data into a vector based data. The parameter set can be varied for an entire classifier category. It can discard the web page if it does not meet a minimum parameter criteria.

3. Classifier: This program will be a multi class classifier designed using the support vector machine model. The classifier will train on data created for each class. The classifier will classify all the newly reported web pages.

4. Portal server: This program would display the results in a hierarchical knowledge structure. It will let users add web pages to a particular topic.
CHAPTER III

SUPPORT VECTOR MACHINES

SVM was introduced by Vapnik and co-workers [5, 26] as a learning system that use a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory. Since its introduction it has already outperformed most other systems in a wider variety of applications.

Linear SVM Model

The simplest model of support vector machines is the maximal margin classifier. It works only for data which are linearly separable in the feature space and hence cannot be used in many real-world situations. Nonetheless it forms the main building block for the more complex support vector machines.

Suppose we have N training data points \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\) where \(x_i \in \mathbb{R}^d\) and \(y_i \in \pm 1\). \(x_i\) is the input vector of dimension \(d\) and \(y_i\) is the corresponding output which indicates whether the input data is part of the given model or not.

We would like to learn a linear separating hyper plane classifier.

\[ f(x) = \text{sgn}(w \cdot x - b) \]
where $w$ is the weight vector and $b$ is the threshold.

Furthermore, we want this hyper plane to have the maximum separating margin with respect to the two classes. Specifically, we want to find this hyper plane $H : y = w \cdot x - b = 0$ and two hyper planes parallel to it and with equal distances to it,

$$H_1 : y = w \cdot x - b = +1$$

$$H_2 : y = w \cdot x - b = -1$$

with the condition that there are no data points between $H_1$ and $H_2$, and the distance between $H_1$ and $H_2$ is maximized.

For any separating plane $H$ and the corresponding $H_1$ and $H_2$, we can always normalize the coefficients vector $w$ so that $H_1$ will be $y = w \cdot x - b = +1$ and $H_2$ will be $y = w \cdot x - b = -1$.

We want to maximize the distance between $H_1$ and $H_2$. So there will be some positive examples on $H_1$ and some negative examples on $H_2$. These examples are called support vectors because only they participate in the definition of the separating hyper plane, and other examples can be removed and/or moved around as long as they do not cross the planes $H_1$ and $H_2$.

Recall that in 2-D, the distance from a point $(x_o,y_o)$ to a line $Ax_o + By_o + C = 0$ is

$$\frac{|Ax_o + By_o + C|}{\sqrt{A^2 + B^2}}.$$ 

Similarly the distance of a point on $H_1$ to $H : w \cdot x - b = 0$ is

$$\frac{|w \cdot x - b|}{\|w\|} = \frac{1}{\|w\|}$$

and the distance between $H_1$ and $H_2$ is $\frac{2}{\|w\|}$. So, in order to maximize the distance, we should minimize $\|w\| = w^T w$ with the condition that there are no data points between $H_1$ and $H_2$.

$$w \cdot x - b \geq +1 \text{ for positive examples } y_i = +1.$$ 

$$w \cdot x - b \leq -1 \text{ for negative examples } y_i = -1.$$ 

These two conditions can be combined into
\[ y_i(w \cdot x - b) \geq 1 \]

So our problem can be formulated as
\[ \min_{w,b} \frac{1}{2} w^T w \text{ subject to } y_i(w \cdot x - b) \geq 1, i = 1, \ldots, n \]

This is a convex quadratic problem. A real-valued function \( f(w) \) is called convex for \( w \in \mathbb{R}^n \), if \( \forall w, u \in \mathbb{R}^n \) and for any \( \theta \in (0,1) \),

\[ f(\theta w + (1 - \theta)u) \leq \theta f(w) + (1 - \theta)f(u) \]

**Lagrangian Theory and Karush-Kuhn-Tucker theory**

The purpose of Lagrangian theory is to characterize the solutions of an optimization problem initially when there are no inequality constraints. The main concepts of this theory are the Lagrange multipliers and the Lagrangian function. Kuhn and Tucker extended the method to allow inequality constraints in what is known as Kuhn-Tucker theory. These will be needful to develop efficient solutions for the task of optimizing SVMs.

In constrained problems as the SVM problem, one needs to define a function, known as the Lagrangian, that incorporates information about the objective function and the constraints, and whose stationary can be used to detect solutions. Precisely, the Lagrangian is defined as the objective function plus a linear combination of the constraints, where the coefficients of the combination are called the Lagrange multipliers.

Given an optimization problem

minimize \( f(w) \) where \( f \) is convex

subject to \( g_i(w) \leq 0, i = 1, \ldots, k \)

We define the Lagrangian function as
\[ L(w, \alpha) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) \]

As per Kuhn-Tucker theorem, for a normal point \( w^* \) to be optimum the following conditions should be satisfied

\[ \frac{\partial L}{\partial w} = 0 \]

\[ \alpha_i g_i(w^*) = 0, \quad i = 1, \ldots, k \]

\[ g_i(w^*) \leq 0, \quad i = 1, \ldots, k \]

\[ \alpha_i^* \geq 0, \quad i = 1, \ldots, k \]

The second condition is known as karush-kuhn-Tucker complementary condition. It implies that for active constraints, \( \alpha_i^* \geq 0 \), whereas for inactive constraints \( \alpha_i^* = 0 \)

Introducing Lagrange multipliers to our linear SVM problem,

\[
\min_{w,b} \frac{1}{2} w^T w \text{ subject to } y_i (w \cdot x - b) \geq 1, \quad i = 1, \ldots, n \text{ or } \\
\min_{w,b} \frac{1}{2} w^T w \text{ subject to } -y_i (w \cdot x - b) + 1 \leq 0, \quad i = 1, \ldots, n
\]

we have the following Lagrangian

\[ L(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i y_i (w \cdot x_i - b) + \sum_{i=1}^{n} \alpha_i \] (1)

Lagrangian treatment of convex optimization problems leads to an alternative dual description, which often turns out to be easier to solve than the primal problem since handling inequality constraints directly is difficult. The dual problem is based on the idea that the dual variables (Lagrange multipliers) are the fundamental unknowns.
of the problem. We can transform the primal into a dual by simply setting to zero the
derivatives of the Lagrangian with respect to the primal variables, and substituting
the relations so obtained back into the Lagrangian, hence removing the dependence
on the primal variables. The dual representations in SVM not only allows us to work
in high dimensional spaces but also paves the way for algorithmic techniques derived
from optimization theory.

We can form dual problem by maximizing Lagrangian in 1 with respect to $\alpha$
subject to the constraints that the gradient of $L(w, b, \alpha)$ with respect to the primal
variables $w$ and $b$ vanish

$$
\frac{\partial L}{\partial w} = 0 \quad (2)
$$

$$
\frac{\partial L}{\partial b} = 0 \quad (3)
$$

and that $\alpha \geq 0$.

From Equations 2 and 3, we have

$$
w = \sum_{i=1}^{n} \alpha_i y_i x_i
$$

$$
\sum_{i=1}^{n} \alpha_i y_i = 0
$$

Substitute them into $L(w, \alpha, b)$, we have

$$
L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i x_j
$$

in which the primals are eliminated.

When we solve $\alpha_i$, we can get $w = \sum_{i=1}^{n} \alpha_i y_i x_i$

For $w$ to be optimal, we also have to satisfy the Karush-Kuhn-Tucker comple-
mentary condition which is
\[ \alpha_i y_i (w \cdot x_i - b) = 0, \ i = 1, \ldots, n \]

**Non-linear SVM**

If the surface separating the two classes is not linear then we can transform the data points to another high dimensional space such that the data points will be linearly separable. Let the transformation be \( \Phi() \). In the high dimensional space, we solve

\[
L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \Phi(x_i) \cdot \Phi(x_j)
\]

If we have a way of computing the inner product \( \langle \Phi(x_i) \cdot \Phi(x_j) \rangle \) in feature space directly as a function of the original input points, it becomes possible to merge the two steps needed to build a non-linear learning machine. We call such a direct computation method a kernel function \( K(x, z) = \langle \Phi(x_i) \cdot \Phi(x_j) \rangle \). Defining a kernel function for an input space is frequently more natural than creating a complicated feature space. The kernel function must be symmetric and the matrix \( K = (K(x_i, x_j))_{i,j=1}^{n} \) is positive semi-definite (has non-negative eigenvalues).

**Imperfect Separation**

The other direction to extend SVM is allow for imperfect separation that is we do not strictly enforce that there be no data points between \( H_1 \) and \( H_2 \), but we definitely want to penalize the data points that cross the boundaries. We introduce non-negative slack variables \( \xi_i \geq 0 \), so that

\[
w \cdot x_i - b \geq +1 - \xi_i \text{ for } y_i = +1
\]
\[ w \cdot x_i - b \geq -1 + \xi_i \text{ for } y_i = -1 \]
\[ \xi_i \geq 0 \quad \forall i \]

and we add to the objective function a penalizing term derived from the following theorem,

Consider thresholding real-valued linear functions \( L \) with unit weight vectors on an inner product space \( X \) and fix \( \gamma \in \mathbb{R}^+ \). There is a constant \( C \), such that for any probability distribution \( D \) on \( X \times \{-1, 1\} \) with support in a ball of radius \( R \) around the origin, with probability \( 1 - \delta \) over \( l \) random examples \( S \), any hypothesis \( f \in L \) has error no more than

\[
\text{err}_D(f) \leq \frac{c}{l} \left( R^2 + \frac{||\xi||^2}{\gamma^2} \log^2 l + \log \frac{1}{\delta} \right)
\]

where \( \xi = \xi(f, S, \gamma) \) is the margin slack vector with respect to \( f \) and \( \gamma \).

This theorem bounds the generalization error in terms of the 2-norm of the margin slack vector, the so-called 2-norm soft margin, which contains the \( \xi_i \) scaled by the norm of the weight vector \( w \). Hence the equivalent expression on which the generalization depends is

\[
\frac{R^2 + ||\xi||^2}{\gamma^2} = ||w||^2 \left( R^2 + \frac{||\xi||^2}{||w||^2} \right) \text{ as } \gamma = 1/||w||
\]

\[
= ||w||^2 R^2 + ||\xi||^2
\]

suggesting that an optimal choice for \( C \) in the objective function of the resulting optimization problem should be \( R^{-2} \):

\[
\min_{\xi, b, w} \frac{1}{2} w^T w + C \left( \sum_{i=1}^{n} \xi_i \right)^2
\]

For the 1-norm margin optimization problem the objective function is

\[
\min_{\xi, b, w} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i
\]

subject to \( y_i(w^T x_i - b) + \xi_i - 1 \geq 0, \quad 1 \leq i \leq n \)
\[ \xi_i \geq 0, \quad 1 \leq i \leq n \]

Introducing Lagrange multipliers \( \alpha, \beta \), the Lagrangian is
\[ L(w, b, \xi, \alpha, \beta) = \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i [y_i(w^T x_i - b) + \xi_i - 1] - \sum_{i=1}^{n} \mu_i \xi_i \]

Let’s form the dual problem as done for the linear SVM by differentiating with respect to \( w, \xi \) and \( b \) we get is

\[
\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0
\]

\[
\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0,
\]

\[
\frac{\partial L}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0
\]

Substituting the relations obtained into the primal; we obtain the following dual objective function

\[
\text{maximize}_\alpha L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i \cdot x_j
\]

The KKT optimality conditions of the primal problems are

\[
\alpha_i \left[ y_i(w^T x_i - b) + \xi_i - 1 \right] = 0, \quad 1 \leq i \leq n
\] (4)

\[
\mu_i \xi_i = 0, \quad 1 \leq i \leq n
\] (5)

The constraint \( C - \alpha_i - \mu_i = 0 \) together with \( \mu_i \geq 0 \), enforces \( \alpha_i \leq C \), as \( \xi_i \neq 0 \) only if \( r_i = 0 \) and therefore \( \alpha_i = C \). Therefore \( 0 \leq \alpha_i \leq C \). Depending on the value of \( \alpha_i \), we have three cases to consider

1. If \( \alpha_i = 0 \), then \( \mu_i = C - \alpha_i = C > 0 \). From Equation 5, \( \xi_i = 0 \). So we have
   \[ y_i(w^T x_i - b) - 1 \geq 0. \]

2. If \( 0 < \alpha_i < C \), from Equation 4, we have
   \[ y_i(w^T x_i - b) + \xi_i - 1 = 0. \] Note that \( \mu_i = C - \alpha_i > 0 \), so \( \xi_i = 0 \) (equation 5).
   Substituting we have \( y_i(w^T x_i - b) - 1 = 0 \).

3. If \( \alpha_i = C \), then from Equation 4, we have \( y_i(w^T x_i - b) + \xi_i - 1 = 0 \). Note that \( \mu_i = C - \alpha_i = 0 \), we have \( \xi_i \geq 0 \). So \( y_i(w^T x_i - b) - 1 \leq 0 \).
The quantity \( y_i(w^T x_i - b) - 1 \) can be computed as \( R_i = y_i(w^T x_i - b) - y_i^2 = y_i(w^T x_i - b - y_i) = y_i E_i \) where \( E_i = w^T x_i - b - y_i = u_i - y_i \) is the prediction error.

To summarize the KKT condition implies:

\[ \alpha_i = 0 \Rightarrow R_i \geq 0, \]
\[ 0 < \alpha_i < C \Rightarrow R_i \approx 0, \]
\[ \alpha_i = C \Rightarrow R_i \leq 0. \]

In the following two cases, the KKT condition is violated:

\[ \alpha_i < C \] (this conditions implies that \( \alpha_i = 0 \) ) and \( R_i < 0, \)
\[ \alpha_i > 0 \] (this conditions implies that \( \alpha_i = C \))and \( R_i > 0. \)

**Checking KKT conditions without using threshold b**

As the dual problem does not solve for the threshold \( b \) directly, it would be beneficial to check the KKT condition without using threshold \( b \). This technique is due to keerthi et al [15].

The quantity \( y_i(w^T x_i - b) - 1 \) (which must \( \geq 0 \) for all \( i \) if the KKT condition is satisfied) can be written as

\[ y_i(w^T x_i - b) - 1 \]
\[ = y_i(w^T x_i - b) - y_i^2 \]
\[ = y_i(w^T x_i - b - y_i) \]
\[ = y_i(F_i - b) \]

where \( F_i = w^T x_i - y_i \)

Note for \( E_i = F_i - b \), we have \( E_i - E_j = F_i - F_j \).

This notation is useful because the KKT Conditions

\[ \alpha_i = 0 \Rightarrow y_i(F_i - b) \geq 0, \]
\[ 0 < \alpha_i < C \Rightarrow y_i(F_i - b) \approx 0, \]
\( \alpha_i = C \Rightarrow y_i (F_i - b) \leq 0. \)

can be written as
\( i \in I_0 \cup I_1 \cup I_2 \Rightarrow F_i \geq b \)
\( i \in I_0 \cup I_3 \cup I_4 \Rightarrow F_i \leq b, \)

where
\( I_0 \equiv i : 0 < \alpha_i < C \)
\( I_1 \equiv i : y_i = +1, \alpha_i = 0 \)
\( I_2 \equiv i : y_i = -1, \alpha_i = C \)
\( I_3 \equiv i : y_i = +1, \alpha_i = C \)
\( I_4 \equiv i : y_i = -1, \alpha_i = 0 \)

So that \( \forall i \in I_0 \cup I_1 \cup I_2, \) and \( \forall j \in I_0 \cup I_3 \cup I_4, \) we should have \( F_i \geq F_j, \) if KKT condition is satisfied.

To check if this condition holds, we define
\( b_{up} = \min F_i : i \in I_0 \cup I_1 \cup I_2 \)
\( b_{low} = \max F_i : i \in I_0 \cup I_3 \cup I_4. \)

The KKT condition implies \( b_{up} \geq b_{low}, \) and similarly, \( \forall i \in I_0 \cup I_1 \cup I_2, F_i \geq b_{low}, \) and \( \forall i \in I_0 \cup I_3 \cup I_4, F_i \leq b_{up}. \)

These comparisons do not use the threshold \( b. \)

As an added benefit, given the first \( \alpha_i, \) these comparisons automatically finds the second \( \alpha_i \) for joint optimization in SMO.
CHAPTER IV

OPTIMIZATION METHODS

To solve the SVM problem one has to solve the (convex) quadratic programming (QP) problem under the constraints. As the objective function is convex every local maximum is already a global maximum. However, there can be several optimal solutions (in terms of the variables $\alpha_i$) which might lead to different testing performances.

The structure of the SVM optimization problem allows to derive specially tailored algorithms which allow for fast convergence with small memory requirements even on large problems. Following three approaches are popular methods

**Chunking**

A key observation in solving large scale SVM problems is the sparsity of the solution $\alpha$. Depending on the problem, many of the optimal $\alpha_i$ will either be zero or on the upper bound $C$. At every step chunking solves the problem containing all non-zero $\alpha_i$ plus some of the $\alpha_i$ violating the KKT conditions. The size of this problem varies but is finally equal to the number of non-zero coefficients. While this technique is suitable for fairly large problems it is still limited by the maximal number of support vectors that one can handle and it still requires a quadratic optimizer to solve the sequence of smaller problems.
Decomposition Methods

This is similar in spirit to chunking as they solve a sequence of small QPs as well. But here the size of the sub problems is fixes. They are based on the observations that a sequence of QPs which at least always contain one sample violating the KKT conditions will eventually converge to the optimal solution. It was suggested to keep the size of the subproblems fixed and to add and remove one sample in each iteration. This allows the training of arbitrary large data sets. In practice, however, the convergence of such an approach is very slow. Practical implementations use sophisticated heuristics to select several patterns to add and remove from the subproblem plus efficient caching methods. They usually achieve fast convergence even on large datasets with up to several thousands of support vectors.

Sequential Minimal Optimization

This method is an extreme case of decomposition methods. In each iteration it solves a quadratic problem of size two. This can be done analytically and thus no quadratic optimizer is required. Here the main problem is to chose a good pair of variables to optimize in each iteration. The original heuristics presented are based on the KKT conditions and there has been some work to improve them. It is a simple method that quickly solves the SVM QP problem without any extra matrix storage and without invoking an iterative numerical routine for each sub-problem.
CHAPTER V

SEQUENTIAL MINIMAL OPTIMIZATION

Osuna et al [18] showed that the large QP problem can be broken down into a series of smaller QP sub-problems. As long as at least one example that violates the KKT condition is added to the examples for the previous sub-problem, each step reduces the overall objective function and maintains a feasible point that obeys all of the constraints. Therefore, a sequence of QP sub-problems that always add at least one violator will asymptotically converge. Osuna et al suggest keeping a constant size matrix for every QP sub-problem, which implies adding and deleting the same number of examples at every step. Using a constant-size matrix allows the training of arbitrarily sized data sets. The algorithm given in Osuna’s paper suggests adding one example and subtracting one example at every step.

SMO introduced by John Platt [19] decomposes the overall QP problem into QP sub-problems similar to Osuna’s method. For the standard SVM QP problem, the smallest possible optimization problem involves two Lagrange multipliers because the Lagrange multipliers must obey a linear equality constraint. At every step, SMO chooses two Lagrange multipliers to jointly optimize, finds the optimal valued for these multipliers, and updates the SVM to reflect the new optimal value.

Because there are only two multipliers, the constraints can be displayed in two dimensions. The bound constraints cause the Lagrange multipliers to lie within a box, while the linear equality constraint causes the Lagrange multipliers to lie on a
diagonal line. Thus, the constrained maximum of the objective function must lie on a diagonal line segment. This constraint explains why two is the minimum number of Lagrange multipliers that can be optimized: if SMO optimized only one multiplier, it could not fulfill the linear equality constraint at every step.

The ends of the diagonal line segment can be expressed quite simply. Without loss of generality, the algorithm first computes the second Lagrange multiplier \( \alpha_2 \) and computes the ends of the diagonal line segment in terms of \( \alpha_2 \).

Because \( \sum_{i=1}^{N} y_i \alpha_i = 0 \), we have
\[
y_1 \alpha_1 + y_2 \alpha_2 = y_1 \alpha_1^{old} + y_2 \alpha_2^{old}
\]

Multiplying left hand side of above equation by \( y_1 \), we have
\[
y_1^2 \alpha_1 + y_1 y_2 \alpha_2 = Const. \text{ Since } y_1^2 = 1, \text{ we have}
\]
\[
\alpha_1 + s \alpha_2 = \gamma \text{ or } \alpha_1 = \gamma - s \alpha_2 \text{ where } s = y_1 y_2 \text{ and } \gamma \text{ is a constant which is}
\]
\[
\gamma = \alpha_1 + s \alpha_2 = \alpha_1^{old} + s \alpha_2^{old}
\]

Fixing the other \( \alpha_i \)'s, the objective function can be written as

\[
L = \alpha_1 + \alpha_2 + Const - \frac{1}{2}(y_1 y_2 x_1^T x_1 \alpha_1 + y_2 y_2 x_2^T x_2 \alpha_2 + 2 y_1 y_2 x_1^T x_2 \alpha_1 \alpha_2 + 2 \sum_{i=3}^{N} \alpha_i y_i x_i^T) (y_1 x_1 \alpha_1 + y_2 x_2 \alpha_2) + Const.)
\]

Let \( K_{11} = x_1^T x_1 \), \( K_{22} = x_2^T x_2 \), \( K_{12} = x_1^T x_2 \), and

\[
v_j = \sum_{i=3}^{N} \alpha_i y_i x_i^T x_j = \sum_{i=1}^{N} \alpha_i y_i x_i^T x_j - \alpha_1^{old} y_1 x_1^T x_j - \alpha_2^{old} y_2 x_2^T x_j
\]
\[
= x_j^T w^{old} - \alpha_1^{old} y_1 x_1^T x_j - \alpha_2^{old} y_2 x_2^T x_j \text{ as } w = \sum_{i=1}^{N} \alpha_i y_i x_i
\]
\[
= (x_j^T w^{old} - b^{old}) + b^{old} - \alpha_1^{old} y_1 x_1^T x_j - \alpha_2^{old} y_2 x_2^T x_j
\]
\[
= v_j^{old} + b^{old} - \alpha_1^{old} y_1 x_1^T x_j - \alpha_2^{old} y_2 x_2^T x_j,
\]

where \( v_j^{old} = x_j^T w^{old} - b^{old} \) is the output of \( x_j \) under old parameters.
\[ L_D = \alpha_1 + \alpha_2 - \frac{1}{2} \left( K_{11}\alpha_1^2 + K_{22}\alpha_2^2 + 2sK_{12}\alpha_1\alpha_2 + 2y_1v_1\alpha_1 + 2y_2v_2\alpha_2 \right) \\
+\text{Const.} \]

Substituting \( \alpha_1 = \gamma - s\alpha_2 \) and \( y_1\alpha_1 = \text{Const} - y_2\alpha_2 \), we have

\[ = \gamma - s\alpha_2 + \alpha_2 - \frac{1}{2} (K_{11}(\gamma - s\alpha_2)^2 + K_{22}\alpha_2^2 + 2sK_{12}(\gamma - s\alpha_2)\alpha_2 + 2v_1(\text{Const} - y_2\alpha_2) + 2y_2v_2\alpha_2) + \text{Const.} \]

\[ = (1 - s)\alpha_2 - \frac{1}{2} K_{11}(\gamma - s\alpha_2)^2 - \frac{1}{2} K_{22}\alpha_2^2 - sK_{12}(\gamma - s\alpha_2)\alpha_2 \\
+ y_2v_1\alpha_2 - y_2v_2\alpha_2 + \text{Const.} \]

as \( \gamma \) is constant

\[ = (1 - s)\alpha_2 - \frac{1}{2} K_{11}\gamma^2 + sK_{11}\gamma\alpha_2 - \frac{1}{2} K_{11}s^2\alpha_2^2 - \frac{1}{2} K_{22}\alpha_2^2 - sK_{12}\gamma\alpha_2 + s^2K_{12}\alpha_2^2 + y_2v_1\alpha_2 - y_2v_2\alpha_2 + \text{Const.} \]

\[ = (1 - s)\alpha_2 + sK_{11}\gamma\alpha_2 - \frac{1}{2} K_{11}\alpha_2^2 - \frac{1}{2} K_{22}\alpha_2^2 - sK_{12}\gamma\alpha_2 + K_{12}\alpha_2^2 + y_2v_1\alpha_2 - y_2v_2\alpha_2 + \text{Const} \]

\[ = \left( -\frac{1}{2} K_{11} - \frac{1}{2} K_{22} + K_{12}\right)\alpha_2^2 + (1 - s + sK_{11}\gamma - sK_{12}\gamma + y_2v_1 - y_2v_2)\alpha_2 \\
+\text{Const} \]

Let \( \eta = 2K_{12} - K_{11} - K_{12} \).

The coefficient of \( \alpha_2 \) is

\( (1 - s + sK_{11}\gamma - sK_{12}\gamma + y_2v_1 - y_2v_2) \)
\[ L_D = \frac{1}{2} \eta \alpha^2 + (y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta \alpha_2^{\text{old}}) \alpha_2 + \text{Const.} \]

The first and second derivatives are
\[
\frac{dL_D}{d\alpha_2} = \eta \alpha_2 + (y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta \alpha_2^{\text{old}}),
\]
\[
\frac{d^2L_D}{d\alpha_2^2} = \eta
\]
Note that \( \eta = 2K_{12} - K_{11} - K_{12} \leq 0 \).

**Proof:** Let \( K_{11} = x_1^T x_1, K_{12} = x_1^T x_2, K_{22} = x_2^T x_2 \).
Then \( \eta = -(x_2 - x_1)^T (x_2 - x_1) = -\|x_2 - x_1\|^2 \leq 0 \).
Let \( \frac{dL_D}{d\alpha_2} = 0 \), and we have
\[
\alpha_2^{\text{new}} = -\frac{y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta \alpha_2^{\text{old}}}{\eta}
\]
\[= \alpha_2^{\text{old}} + \frac{y_2(E_1^{\text{old}} - E_2^{\text{old}})}{\eta} \]
If $\eta < 0$, the above eqn gives us the unconstrained maximum point $\alpha_{2}^{\text{new}}$. It must be checked against the feasible range.

Let $s = y_1y_2$, and $\gamma = \alpha_1^{\text{old}} + s\alpha_2^{\text{old}}$. The range of $\alpha_2$ is determined as follows:

- If $s=1$, then $\alpha_1 + \alpha_2 = \gamma$
  
  * if $\gamma > C$, then $\max \alpha_2 = C$ and $\min \alpha_2 = \gamma - C$
  * if $\gamma < C$, then $\max \alpha_2 = \gamma$ and $\min \alpha_2 = 0$

- If $s=-1$, then $\alpha_1 - \alpha_2 = \gamma$

  * if $\gamma > 0$, then $\max \alpha_2 = C - \gamma$ and $\min \alpha_2 = 0$
  * if $\gamma < 0$, then $\max \alpha_2 = C$ and $\min \alpha_2 = -\gamma$

Let the minimum feasible value of $\alpha_2$ be $L$, maximum be $H$. Then

$$\alpha_{2}^{\text{new,clipped}} = \begin{cases} H, & \text{if } H < \alpha_{2}^{\text{new}} \\ \alpha_{2}^{\text{new}}, & \text{if } L \leq \alpha_{2}^{\text{new}} \leq H \\ L, & \text{if } \alpha_{2}^{\text{new}} < L. \end{cases}$$

To summarize, given $\alpha_1, \alpha_2$ ($y_1, y_2, K_{11}, K_{12}, K_{22}, E_2^{\text{old}} - E_1^{\text{old}}$), we can optimize the two $\alpha$’s by the following procedure

1. $\eta = 2K_{12} - K_{11} - K_{22}$
2. If $\eta < 0$,
   
   (a) $\Delta\alpha_2 = \frac{y_2(E_2^{\text{old}} - E_1^{\text{old}})}{\eta}$ and clip the solution within the feasible region. Then $\Delta\alpha_1 = -s\Delta\alpha_2$. 
(3) If \( \eta = 0 \), we need to evaluate the objective function at the two endpoints and set \( \alpha_{new} \) to be the one with larger objective function value. The objective function is

\[
L_D = \frac{1}{2} \eta \alpha^2 + (y_2(E_{1_{old}} - E_{2_{old}}) - \eta \alpha_{old}^2) \alpha_2 + Const.
\]

When \( \alpha_1, \alpha_2 \) are changed by \( \Delta \alpha_1, \Delta \alpha_2 \), we can update \( E_i \)'s, \( F_i \)'s, w(for linear kernel), and \( b \). Let \( E(x, y) \) be the prediction error on \( (x, y) \):

\[
E(x, y) = \sum_{i=1}^{n} \alpha_i y_i x_i^T x_i - b - y,
\]

The change in \( E \) is

\[
\Delta E(x, y) = \Delta \alpha_1 y_1 x_1^T x + \Delta \alpha_2 y_2 x_2^T x - \Delta b
\]

The change in the threshold can be computed by forcing \( E_{1_{new}} = 0 \) if \( 0 < \alpha_{1_{new}} < C \) (or \( E_{2_{new}} = 0 \) if \( 0 < \alpha_{2_{new}} < C \)). From

\[
0 = E(x, y)^{new}
\]

\[
= E(x, y)^{old} + \Delta E(x, y)
\]

\[
= E(x, y)^{old} + \Delta \alpha_1 y_1 x_1^T x + \Delta \alpha_2 y_2 x_2^T x - \Delta b
\]

we have

\[
\Delta b = E(x, y)^{old} + \Delta \alpha_1 y_1 x_1^T x + \Delta \alpha_2 y_2 x_2^T x
\]

If \( \alpha_1 = 0 \), we can only say \( y_1 E_{1_{new}} \geq 0 \); similarly, if \( \alpha_1 = C \), we have \( y_1 E_{2_{new}} \leq 0 \). If both \( \alpha_1 \) and \( \alpha_2 \) take values 0 or \( C \), the original SMO algorithm computes two values of the new \( b \) for \( \alpha_1 \) and \( \alpha_2 \) and takes the average. This is regarded as problematic by keerthi et al [15].

Similarly, from

\[
F(x, y) = \sum_{i=1}^{n} \alpha_i y_i x_i^T x - y
\]

we have

\[
\Delta F(x, y) = \Delta \alpha_1 y_1 x_1^T x + \Delta \alpha_2 y_2 x_2^T x
\]
For the weight vector of linear kernels,
\[ w = \sum_{i=1}^{n} \alpha_i y_i x_i \]
\[ \Delta w = \Delta \alpha_1 y_1 x_1 + \Delta \alpha_2 y_2 x_2 \]

The heuristics for picking two \( \alpha_i \)'s for optimization in the original SMO paper are as follows:

- The outer loop selects the first \( \alpha_i \), the inner loop selects the second \( \alpha_i \) that maximizes \( |E_2 - E_1| \)

- The outer loop alternates between one sweep through all examples and as many sweeps as possible through the non-boundary examples (those with \( 0 < \alpha_i < C \)), selecting the example that violates the KKT condition.

- Given the first \( \alpha_i \), the inner loop looks for a non-boundary that maximizes \( |E_2 - E_1| \). If this does not make progress, it starts a sequential scan through the non-boundary examples, starting at a random position; if this fails too, it starts a sequential scan through all the examples, also starting at a random position.

Because the algorithm spends most of the time adjusting the non-boundary examples, the \( E_i \)'s of these examples are cached.

The improvement proposed in Keerthi et al [15] avoids the use of the threshold \( b \) in checking the KKT condition, and compares two \( F_i \)'s, which also automatically selects the second \( \alpha_i \) for joint optimization. There are two variations when the outer loop deals only with the non-boundary examples.

- The first \( \alpha \) is selected sequentially from all the non-boundary examples. If the first \( \alpha_i \) violates the KKT condition when compared with \( \alpha_i \) with \( F_j = b_{low} \), or \( F_j = b_{up} \), then select \( \alpha_i \) as the second \( \alpha \).
• The two $\alpha$’s are also those with $F_i = b_{low}$ or $F_i = b_{up}$

After a successful step using a pair of indices, $(i_2, i_1)$, let $I = I_o \cup i_1, i_2$. We claim that each of the two sets, $I \cap (I_o \cup I_1 \cup I_2)$ and $I \cap (I_o \cup I_3 \cup I_4)$ is non-empty, hence we can compute partial $b_{low}$ and $b_{up}$ from the two sets.
CHAPTER VI

MULTI-CLASS CLASSIFICATION

Support vector machines are binary classifiers but it can be extended to classify many classes. Several methods have been proposed where typically a multi-class classifier is constructed by combining several binary classifiers or by directly considering all data in one optimization formulation. A general framework for solving multi-class problems using a coding matrix is explained in [1, 8]. Some of the practical methods using binary classifiers are one-against-all, one-against-one and DDAG.

The one-against-all method constructs a SVM model for each class with positive training data of that particular class along with the training data of other classes (negative). The number of classifiers are same as the number of classes. For each classifier, a threshold value is set. When running test data through the classifier, only those data whose output is above the threshold are classified into that category.

The one-against-one method constructs k(k-1)/2 classifiers where k is the number of classes where each one is trained on data from two classes.

Decision Directed Acyclic Graph [20] is similar to one-against-one method as it trains k(k-1)/2 classifiers but the testing is done differently. In one-against-one model the testing is done against all the k(k-1)/2 models and a vote is registered for the winning/preferred class. The class that gets the maximum vote wins.

The DDAG is a tree containing contains k(k-1)/2 nodes and k leaves, each with an associated one-against-one classifier. The testing is done against the root node
and then against the node that contains the preferred class and so on until we the leaf. So n-1 decision nodes will be visited in order to derive an answer. The DDAG [3, 4] has some analysis of generalization established unlike the other two methods. DDAG algorithm is much faster than the other two methods in training and testing times with accuracies comparable to the two methods. DDAG cannot be effectively used when the data belongs to more than one class. For this case, one-against-all method is effective. One-against-all method is widely used in practice and found to offer good accuracies. Error correcting output codes (ECOC) classifiers proposed in [8] have been experimented on standard text collections in [22] and found that SVMs with one-against-all method had the same accuracy as the ECOC methods.
CHAPTER VII

INPUT REPRESENTATION AND FEATURE SELECTION

The input to support vector machine is a vector representation of the web page. Simple representation of the web page text is found to be inadequate as they have sparse information about the class they belong to. Research shows using hyperlinks to the web page (or citations) along with web page offers good results [9]. Text from web page along with the text around the hyperlinks and the anchor texts describing the hyperlink have been found to offer good accuracy. For each web page, 10 web pages containing a hyperlink to it are obtained and about 25 words around the hyperlinks along with anchor text is extracted.

Simple representation of the web document as a bag of words works well over complex representation like factoids, noun phrases and multi word dictionary entries[25].

Several techniques are used in information retrieval to reduce the number of text features as it can run in thousands [10, 23]. Support vector machines are independent of the dimensionality of the feature space but it helps to reduce the features as it saves space and memory requirements. Yang and Pedersen[27] compare a number of methods for feature selection. Document frequency and information gain have been used which have shown excellent results in their comparisons. Document frequency is the number of times a word occurs in a document.

Information gain is the number of bits of information obtained for category prediction by knowing the presence or absence of a term in a document. Information
The information gain (IG) of a word \( w \) is

\[
IG(w) = - \sum_{i=1}^{m} P(c_i) \log P(c_i) + P(w) \sum_{i=1}^{m} P(c_i|w) \log P(c_i|w)
+ P(\overline{w}) \sum_{i=1}^{m} P(c_i|\overline{w}) \log P(c_i|\overline{w})
\]

where \( m \) is the number of classes.

\( P(c_i) \) is the percentage of documents that belong to class \( i \).

\( P(c_i|w) \) is the percentage of documents (which contain the word \( w \)) that belong to \( i \).

\( P(c_i|\overline{w}) \) is the percentage of documents (which do not contain the word \( w \)) that belong to \( i \).

\( P(w) \) is the percentage of documents that contain the word \( w \).

\( P(\overline{w}) \) is the percentage of documents that do not contain the word \( w \).

A stop word list is also used to remove stop words like "a", "the",\ldots from the features. Words are stemmed using Porter algorithm. Also the words are not case sensitive.
CHAPTER VIII

WEB PORTAL SYSTEM

The web portal model consists of a crawler, data converter, classifier(support vector machine), database and web server. The crawler fetches web pages which are converted into vector format by the data converter and fed into the trained classifier for classification. All information is stored in database and the actual data are stored in files. The classified web links are displayed on the web server. The web portal system is shown in figure 1.

Functions of these systems are explained below

Focused Crawler

The function of the crawler is to start from a given web link and fetch the corresponding web page and obtain web links in that web page. This process is done recursively. Information about the searched web links are saved in the database and the actual web page is saved in files under a specified directory. In the database additional information such as the url of web page, the stored file name of the web page, title and description of the web page is stored.

When the crawler is run, it loads the existing links already searched from the database. In the database, all irrelevant (non technical) web links are also listed. It
Figure 1: Web Portal System Overview
maintains three lists discard list, urls-fetched list and urls-to-fetch list.Urls-to-fetch list initially has only the web link that the crawler had been given when it was started.

The crawler gets a link from urls-to-fetch list and fetches the web page, title and description of the web link and stores them. It also gets the web links listed in this web page(source link). If the listed web links have the same base as the source link, then these links are ignored. Also if the listed web links are images or if they are present in the discard list, then they are ignored. If it is listed in urls-fetched list, then it increases the rank of the web link and also adds the source web link to citation list( this list maintains all the web links that have the current link listed in their web pages). If the listed web links do not satisfy the above conditions then they are added to url-to-search list and their rank is increased and also the source web link is added to their citation list. The above steps are done recursively till the number of fetched urls reach the maximum limit or all links in the urls-to-fetch are obtained.

This program is implemented in Perl using the LWP modules.

**Data Converter**

Data converter's function is to convert the given data(web link) into a vector format readable for the classifier. Its function varies for training and testing data. It does more work for training data as it collects data and does feature selection. All code for data converter are implemented using Perl using the HTML::LinkExtor module.

**Training**

There are many technological fields and each of them has many subcategories. For each category, training data has to be provided. The training data consists of
web links and the categories they belong to.

The web page corresponding to each of these web links are fetched and stored in files under a specified directory. Ten web pages that contain links to these training data web links are then fetched and from them anchor/extended texts are extracted and stored in files under a specified directory. To get these web pages, Google’s api is used. Google limits the number of queries results to 1000 per day so this it has to be done in a batch mode for a number of days depending on the size of the training data. All these information are logged into the database.

Once all training data are fetched, feature selection process is started. The training data are represented as a bag of words and information gain for each word is then calculated. Words less than a certain document frequency can be dropped in this process.

All stop words are removed. Information Gain is calculated for each of the word and feature reduction can be done by selecting only words that are above certain information gain value and these words are stored in the database for testing purposes. The training data can be stored in a vector format either as number of occurrences of these words (sparse) or as a simple presence/absence of these words (sparse binary)

Testing

Each web link has three data sources, the web page, anchor text and extended anchor text. All these three data are combined and represented as a bag of words. Only certain words are extracted which is given by the word list stored in the database during the training period.

Classifier - Support Vector Machine
The classifier gets the training and testing documents from the database and with help of the data converter converts the data into a vector format. It classifies and stores the results back in the database.

The classifier uses the support vector machine learning method and is implemented in C++ using OO concepts. The following features are implemented

- Implements the SMO algorithm with improvements provided by keerthi et al [15]
- Multiclass and Binary SVM models. Training can be done on a particular group of classes or just for one class.
- Supports DDAG and One-against-all multiclass approaches.
- The models can be trained for different parameters like the epsilon, tolerance, kernel type, threshold, etc. The results are stored in the database.
- Supports linear and guassian kernel.
- Supports three types of data- dense, sparse and sparsebinary.
- Code is optimized to use minimal database interactions for speed.
- Testing can be done for an entire group or just for a single class.
- the memory required is minimized during testing. When testing against a group of classes, the classifier models are loaded only once and the test data is tested against that model.

Portal Server
The web portal runs on an apache web server. The web pages are developed in PHP. The links are displayed as per their rank (descending order) under each category. The title and description of the web page is displayed along with the link. Search facility is provided and it searches the given word against the title, description and urls of all categories. On typing a search word, the database is queried for links that contain the search word and the result is displayed on the web page. Users can register themselves and can create/modify a profile. Facility to login is provided for users. Users can suggest web links to be classified for a technical area. Sessions are used to keep a persistent connection with a user.

Database

All the data and results are stored in mysql database. The database consists of the following tables

- **Group**: Stores group name (bioinformatics), parameter list and type of data (sparse, dense,..)

- **Classes**: Stores the class id (unique database id), class name (categories like companies, services..), group name it is associated with, threshold for one-against-all multiclassifying approach

- **Data**: Stores url of the training/testing data, file name which has the web page of the url, whether it is testing/training data, data set name

- **crawlData**: stores data from the crawler which are the url, file name where the web page of the url is stored, title, description, rank, web links which have a link to this url, whether this web link should be discarded.
• Trainset : Contains training data in the vector format along with its class it belongs to, target( +1 if it belongs to the class or -1 if it doesn’t)

• Testset : Contains the testing data in the vector format along with its classes it belongs to, classes that the SVM has assigned, target( +1 if it belongs to the model or -1 if it doesn’t), group name this test data belongs to.

• Result : Contains results from the SVM such as class favored, class , kernel type, constant C (for nonlinear SVM), threshold, epsilon, tolerance, result data, size of result data
CHAPTER IX

TRAINING AND FINE TUNING PERFORMANCE

Performance of classifiers are popularly measured using recall and precision. Given a document as the input to a classifier, and a ranked list of categories as the output, the recall and precision at a particular threshold on this ranked list are defined to be

\[
\text{Recall} = \frac{\text{Categories found and correct}}{\text{total Categories correct}}
\]

\[
\text{Precision} = \frac{\text{Categories found and correct}}{\text{total Categories found}}
\]

For obtaining estimates of precision and recall relative to the whole category set, two different methods may be adopted:

- micro-averaging: precision and recall are obtained by globally summing over all individual decisions.

- macro-averaging: precision and recall are first evaluated locally for each category, and then globally by averaging over the results of the different categories.

Macro-averaging gives an equal weight to the performance on every category regardless how common or rare a category is. Micro-averaging gives an equal weight to the performance on every document thus favoring performance on common categories. Micro-averaging is used in the evaluation of all performance runs.

There is a trade-off between recall and precision. Effectiveness is computed as the breakeven point, the value at which Precision equals Recall. When each one of
the data belongs to only one category then micro-averaging performance on such data
will give an equal Precision and recall value.

Performance of the classifier was primarily checked against four types of input

- Web page - the complete web page of the given url

- Anchor Text - the anchor text of the given url found in ten web pages whose
  url is different from given url

- Extended Text - text surrounding the given url found in ten web pages whose
  url is different from given url

- Combination - all the text mentioned above for a given url

The results were compared to choose the best way to represent the input for the
classifier.

Data

The web portal system performance was tested against an upcoming field Bioin-
formatics. Training data was obtained from google’s Bioinformatics directory. Five
sub categories under Bioinformatics were chosen which covered different areas with
good amount of links listed under them. A category called Discard was added to
account for non-Bioinformatics web pages. 369 web pages were collected for all the
categories. For each of these web pages, anchor text and extended text were extracted
by using the google API. There were 21 web pages which did not have any link to
them so no anchor text and extended text were recovered. The training data was
assumed to be accurate with very low error rate (i.e) incorrectly classified.
There were three web pages that could not be found in the company category which were removed. Six web pages were listed in more than one category and they were all listed in two categories. After all this changes, the total was reduced to 360 web pages.

The data was split as 75% for training and 25% for testing. The testing data was used for fine tuning the SVM parameters and also for the input feature selection.

Unit Testing Classifier

Test data from the UCI ML Repository[2] was used for unit testing the support vector machine as SVMs have been tested against this data extensively. The results were compared and found matching with other similar runs. For testing the binary classifier, tic-tac-toe and adult test data were used. For testing the multiclass classifier, wine and dna test data were used.

Feature and Kernel Selection

Table 1: Number of web pages for each category

<table>
<thead>
<tr>
<th>Category</th>
<th>Number of Web Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Companies</td>
<td>119</td>
</tr>
<tr>
<td>Software</td>
<td>34</td>
</tr>
<tr>
<td>Education</td>
<td>21</td>
</tr>
<tr>
<td>Research</td>
<td>28</td>
</tr>
<tr>
<td>Services</td>
<td>45</td>
</tr>
<tr>
<td>Discard</td>
<td>106</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>369</strong></td>
</tr>
</tbody>
</table>
Number of words generated by changing document frequency for web page, anchor text, extended text and combination of these three text is shown below. Higher document frequency (greater than 2) was not tested as it gave very low word count.

<table>
<thead>
<tr>
<th>Input Type</th>
<th>Words DF=1</th>
<th>Words DF=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Page</td>
<td>12110</td>
<td>3500</td>
</tr>
<tr>
<td>Anchor Text</td>
<td>1822</td>
<td>300</td>
</tr>
<tr>
<td>Extended Text</td>
<td>5091</td>
<td>1513</td>
</tr>
<tr>
<td>Combination</td>
<td>14792</td>
<td>4318</td>
</tr>
</tbody>
</table>

Table 2: Word count for different Document Frequency

To get the optimum features for text, parameters like kernel type, document frequency, Information Gain and data type were changed and performance was checked. Since web page, anchor text and extended text contained all text, testing against one of these input types will be sufficient. Web page was chosen as the input for these performance runs and the classifier used 75% of input data for training and 25% for testing.

Since all the parameters are mutually exclusive, the test run was carried out in two steps to limit the number of test runs. The first test was run against the web page input and the parameters kernel type and information gain were changed. The classifier used DDAG since more than 99% of input belonged to only one category. The accuracies are calculated as the number of correctly assigned test data to total number of test data that are correct (recall).

Linear kernel performed much better than the guassian kernel which is well proved in many tests for text data. Limiting the number of words based on their information gain value improved performance as the limit was decreased but the performance dropped after a certain level. Too few features and too many features decreased the accuracy. Performance was good at an Information gain greater than
0.006829 (5560 words).

Second test was conducted by changing document frequency and data type parameter using linear kernel for the four input types.

From table 4, it can be observed that the performance of sparse binary data type is much better than sparse. Varying document frequency has very slight change in performance. For combination input type, document frequency of 2 has shown better results than document frequency of 1. Various tests conducted such as in [13, 27, 25] have proved that using sparse binary data and feature selection methods like information gain and document frequency to give good performance.

Using just the web document does not give good overall results. There are different approaches to better performance. Joachims et al [14] have tried using citation link along with the web page content. They have used combination of kernels of these two data and found that independent approaches with similar error rate achieve good results.

Glover et al [9] have trained two svms one with web page document data and the other with the extended text. These two SVMs are run against each test data and the two SVM outputs are compared to conclude where the test data should belong.

In this model instead of using two SVMs, web page and extended text have been combined along with the anchor text and the performance was measured. The

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Information Gain(No. Of words)</th>
<th>Accuracy(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>0.0088243(3651)</td>
<td>67</td>
</tr>
<tr>
<td>Linear</td>
<td>0.006829(5560)</td>
<td>71</td>
</tr>
<tr>
<td>Linear</td>
<td>0.004558(6534)</td>
<td>68</td>
</tr>
<tr>
<td>Guassian</td>
<td>0.0088243(3651)</td>
<td>32</td>
</tr>
<tr>
<td>Guassian</td>
<td>0.006829(5560)</td>
<td>32</td>
</tr>
<tr>
<td>Guassian</td>
<td>0.004558(6534)</td>
<td>31</td>
</tr>
</tbody>
</table>

Table 3: Accuracy obtained for Kernel and Information Gain parameters
performance improved considerably for a linear guassian SVM from 63% (extended text) and 68% (web page) to 81% (combination).

Table 5 shows a comparison between the results from the combination input and other input methods on 95 test data. "Both correct" data are data which have the same result as combination and are correctly classified, "Incorrect and correct" data are data which have different result than combination and are incorrectly classified in combination and so on.

The comparison between extended text and web page is shown in table 6.

From table 6 it is clear that there are $45 + 20 + 15 = 80$ correct data between extended text and web page. From table 5, we can see that there are 77 correctly classified using combination input. The combination has correctly classified 77 out of 80 data.

Other tests were also carried by not using stop word list and not stemming words, no significant change in accuracy was noted. SVM parameters Constant C and epsilon were also changed and the best results came out for C=1 and epsilon=0.001
<table>
<thead>
<tr>
<th>Input Type</th>
<th>DF</th>
<th>Data Type</th>
<th>Accuracy(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Page</td>
<td>1</td>
<td>sparse binary</td>
<td>68.4</td>
</tr>
<tr>
<td>Anchor Text</td>
<td>1</td>
<td>sparse binary</td>
<td>47.4</td>
</tr>
<tr>
<td>Extended Text</td>
<td>1</td>
<td>sparse binary</td>
<td>63.1</td>
</tr>
<tr>
<td>Combination</td>
<td>1</td>
<td>sparse binary</td>
<td>81</td>
</tr>
<tr>
<td>Web Page</td>
<td>2</td>
<td>sparse binary</td>
<td>65.3</td>
</tr>
<tr>
<td>Anchor Text</td>
<td>2</td>
<td>sparse binary</td>
<td>44.2</td>
</tr>
<tr>
<td>Extended Text</td>
<td>2</td>
<td>sparse binary</td>
<td>62.1</td>
</tr>
<tr>
<td>Combination</td>
<td>2</td>
<td>sparse binary</td>
<td>83.1</td>
</tr>
<tr>
<td>Web Page</td>
<td>1</td>
<td>sparse</td>
<td>62.1</td>
</tr>
<tr>
<td>Anchor Text</td>
<td>1</td>
<td>sparse</td>
<td>46.3</td>
</tr>
<tr>
<td>Extended Text</td>
<td>1</td>
<td>sparse</td>
<td>60</td>
</tr>
<tr>
<td>Combination</td>
<td>1</td>
<td>sparse</td>
<td>74</td>
</tr>
<tr>
<td>Web Page</td>
<td>2</td>
<td>sparse</td>
<td>61</td>
</tr>
<tr>
<td>Anchor Text</td>
<td>2</td>
<td>sparse</td>
<td>52.6</td>
</tr>
<tr>
<td>Extended Text</td>
<td>2</td>
<td>sparse</td>
<td>58.9</td>
</tr>
<tr>
<td>Combination</td>
<td>2</td>
<td>sparse</td>
<td>77.1</td>
</tr>
</tbody>
</table>

Table 4: Accuracy obtained for various parameters

<table>
<thead>
<tr>
<th>Combination with</th>
<th>Extended Text</th>
<th>Web Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both Correct</td>
<td>55</td>
<td>59</td>
</tr>
<tr>
<td>Both Incorrect</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>Incorrect and correct</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Correct and Incorrect</td>
<td>22</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 5: Web Page Classification Comparison

<table>
<thead>
<tr>
<th>Extended Text with</th>
<th>Web Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both Correct</td>
<td>45</td>
</tr>
<tr>
<td>Both Incorrect</td>
<td>15</td>
</tr>
<tr>
<td>Incorrect and correct</td>
<td>20</td>
</tr>
<tr>
<td>Correct and Incorrect</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 6: Web Page Classification Comparison between extended text and web page
CHAPTER X

PERFORMANCE COMPARISON

Comparison using common dataset

Comparisons were carried on a popular web classification dataset Webkb used in [7, 9, 24]. The dataset contains 4129 pages from four computer science departments and there were 10848 hyperlinks interconnecting them. The data is explained in [6]. Each of the pages were labeled into one of seven classes, department, course, faculty, project, staff, student and other (catch-all-class). For the experiments, the classes other, department and staff were pooled into a single class other. The distribution is given in table 7.

The anchor and extended text information were obtained from the citing pages. Table 8 shows the distribution of no. of web pages that have the same number of links. From the table 75% of web pages have only two citations or less. All training is done using leave-one-university-out cross validation and testing is done on the data of the left out university. All results are microaveraged for four different runs (leaving one of the four university out) for each type of input. The input document words are stemmed and stop words are removed. They are represented as set of words in a sparse binary vector format and input features are reduced by using words that have a minimum document frequency of 2.
Since each web page in the dataset didn’t belong to more than one class, the classifier used DDAG method for multiclass separation as it gave better results than one against all method. All tests were conducted on linear kernel and with similar SVM parameters. The results are tabulated in table 9. No method is clear winner. Web Page performance is better than the anchor and extended text input types which can be attributed to the low number of citations. About 57\% of web pages had only one citation or less. Combination which was expected to perform well actually didn’t come out as overall winner but it was not a loser either. Combination is close to the winning recall/precision scores and never came last. This proves combination has a consistent collective performance when the individual performances fluctuate and the results would have been better if the citing links were more as in the case of Bioinformatics dataset.

A comparison against two other methods which used the Webkb dataset is shown in table 10 for four classes. FOIL-PILFS [7] is a statistical text-learning method with a relational rule learner. SVM(TA) [24] uses support vector machine classifier to classify web pages using the web page text, page title and anchor text. Other methods [11, 9] have been used on a subset of Webkb dataset and hence cannot be compared.

In FOIL-PILFS and SVM(TA) rigorous optimization of threshold of each of the categories have been performed by using portion of the training data called validating

<table>
<thead>
<tr>
<th>University</th>
<th>Course</th>
<th>Faculty</th>
<th>Project</th>
<th>Student</th>
<th>Other</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cornell</td>
<td>44</td>
<td>34</td>
<td>20</td>
<td>128</td>
<td>632</td>
<td>858</td>
</tr>
<tr>
<td>Texas</td>
<td>38</td>
<td>46</td>
<td>18</td>
<td>148</td>
<td>567</td>
<td>817</td>
</tr>
<tr>
<td>Washington</td>
<td>77</td>
<td>31</td>
<td>21</td>
<td>126</td>
<td>945</td>
<td>1200</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>85</td>
<td>42</td>
<td>25</td>
<td>156</td>
<td>946</td>
<td>1254</td>
</tr>
<tr>
<td>Total</td>
<td>244</td>
<td>153</td>
<td>84</td>
<td>558</td>
<td>3090</td>
<td>4129</td>
</tr>
</tbody>
</table>

Table 7: Webkb Dataset
set. In Combination, SVM parameters epsilon and constant C were changed and the best results averaged for the four fold leave-one-university-out cross validation tests. Comparing precision results, combination out performs both methods. While comparing recall results, combination falls behind both methods. This could be attributed to the threshold optimization of other two methods where recall and precision scores are very close. SVM(TA) performance is better than FOIL-PILFs when comparing precision and recall results. SVM(TA) method is very similar to combination, the differences are combination uses extended text in addition to what SVM(TA) uses and SVM(TA) uses $SVM^{light}$ package [12] whose optimization method is different than combination's sequential minimal optimization [19]. Since they are very similar there is no clear winner when comparing SVM(TA) and combination.

**Comparison against other SVM methods**

The combination method is compared to two other SVM techniques used for classifying web pages using citation links.

- Joachims et al [14] used citation link information along with the web page content and they combined the kernels of these two data in one classifier

<table>
<thead>
<tr>
<th>No. Of Links</th>
<th>No. of Web pages</th>
<th>Cumulative Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>92</td>
<td>2.23</td>
</tr>
<tr>
<td>1</td>
<td>2280</td>
<td>57.45</td>
</tr>
<tr>
<td>2</td>
<td>731</td>
<td>75.15</td>
</tr>
<tr>
<td>3</td>
<td>343</td>
<td>83.46</td>
</tr>
<tr>
<td>4</td>
<td>158</td>
<td>87.29</td>
</tr>
<tr>
<td>5</td>
<td>158</td>
<td>91.11</td>
</tr>
<tr>
<td>6-10</td>
<td>245</td>
<td>97.05</td>
</tr>
<tr>
<td>11-178</td>
<td>122</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 8: Webkb Dataset Links Distribution
and found that independent approaches with similar error rate and with low
support vector overlap between the individual kernels achieve good results.
A combination of kernels of web page, extended text and anchor text was
run. A combination of kernels such as the following below is a valid kernel
and the proof is provided in [14]

\[ K(x, z) = \lambda K_1(x, z) + (1 - \lambda) K_2(x, z) \]

For linear kernels and \( \lambda = 0.5 \), we have

\[ x.z = 0.5x_1.z_1 + 0.5x_2.z_2 \]

- Glover et al [9] have trained two svms one with web page data and the other
  with the extended text. These two SVMs were run against each test data and
  the two SVM outputs were compared to conclude where the test data should
  belong. They have used two approaches.

  * One is to use the extended text classifier result except when the result
    lies inside the range (-1,1) (uncertain region), in that case they manually
    classify them.

  * The other approach is to use the extended anchor text output except
    when it is negative and in the uncertain region (>1) then take the
    output of the web page classifier.

<table>
<thead>
<tr>
<th>Method</th>
<th>Course Re</th>
<th>Course Pr</th>
<th>Faculty Re</th>
<th>Faculty Pr</th>
<th>Project Re</th>
<th>Project Pr</th>
<th>Student Re</th>
<th>Student Pr</th>
<th>Other Re</th>
<th>Other Pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anchor Text</td>
<td>.348</td>
<td>.664</td>
<td>.229</td>
<td>.515</td>
<td>.191</td>
<td>.533</td>
<td>.237</td>
<td>.638</td>
<td>.962</td>
<td>.804</td>
</tr>
<tr>
<td>Web Page</td>
<td>.533</td>
<td>.586</td>
<td>.484</td>
<td>.705</td>
<td>.143</td>
<td>.222</td>
<td>.559</td>
<td>.674</td>
<td>.919</td>
<td>.865</td>
</tr>
<tr>
<td>Combination</td>
<td>.488</td>
<td>.567</td>
<td>.353</td>
<td>.720</td>
<td>.155</td>
<td>.317</td>
<td>.473</td>
<td>.680</td>
<td>.935</td>
<td>.846</td>
</tr>
</tbody>
</table>

Table 9: Webkb Dataset Results
Tests were conducted on the Bioinformatics dataset as it had better citation links than the webkb dataset. The first approach of Eric Glover et al was not compared as 93% of the classification output lie in the uncertain region (-1,1). The comparison is tabulated in Table 11.

All runs were profiled using gprof with a max error rate of 3% on a Pentium III 1 Ghz Processor with 256 MB RAM. Anchor text had the shortest run while combination surprisingly took less time than web page maybe because of faster convergence. Combination was faster compared to Eric Glover’s and Kernel Combination classifiers.

Eric Glover classifier time taken is the summation of the time taken by the individual extended and web page text classifiers. Kernel Combination is very similar to the combination method except that in combination features common among the three text are represented only once but in kernel combination they are represented individually which leads to additional time when multiplying two vectors. To reduce the input features, a document frequency of 2 was used. For combination this lead to additional words being selected (around 300) than Kernel combination as all the different text combined in one set leads to higher document frequency. This may have led to the slight increase in performance for the combination than kernel combination.

Clearly combination classifier has performed better than these two methods and also is faster and simpler to implement just one classifier, one combined input dataset and one kernel function. Whereas in Eric Glover we need two classifiers and

<table>
<thead>
<tr>
<th>Method</th>
<th>Course Re</th>
<th>Course Pr</th>
<th>Faculty Re</th>
<th>Faculty Pr</th>
<th>Project Re</th>
<th>Project Pr</th>
<th>Student Re</th>
<th>Student Pr</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOIL-PILFS</td>
<td>.533</td>
<td>.526</td>
<td>.36</td>
<td>.550</td>
<td>.274</td>
<td>.277</td>
<td>.462</td>
<td>.655</td>
</tr>
<tr>
<td>SVM(TA)</td>
<td>.734</td>
<td>.637</td>
<td>.691</td>
<td>.63</td>
<td>.357</td>
<td>.299</td>
<td>.726</td>
<td>.735</td>
</tr>
<tr>
<td>Combination</td>
<td>.444</td>
<td>.778</td>
<td>.263</td>
<td>.781</td>
<td>.104</td>
<td>.454</td>
<td>.475</td>
<td>.828</td>
</tr>
</tbody>
</table>

Table 10: Comparison of different methods
two datasets with a classifier output comparison program and in Kernel combination we need to use one classifier, three datasets and three kernel functions.

<table>
<thead>
<tr>
<th>Method</th>
<th>Run Time Secs</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anchor</td>
<td>12.96</td>
<td>.301</td>
<td>.583</td>
</tr>
<tr>
<td>Extended</td>
<td>90.22</td>
<td>.505</td>
<td>.797</td>
</tr>
<tr>
<td>Web Page</td>
<td>540.34</td>
<td>.591</td>
<td>.743</td>
</tr>
<tr>
<td>Eric Glover</td>
<td>630.56</td>
<td>.742</td>
<td>.750</td>
</tr>
<tr>
<td>Kernel Comb</td>
<td>535.83</td>
<td>.699</td>
<td>.890</td>
</tr>
<tr>
<td>Combination</td>
<td>495.89</td>
<td>.720</td>
<td>.905</td>
</tr>
</tbody>
</table>

Table 11: Comparison of SVM Methods

Kernel combination method is very similar to the combination classifier with a small difference in the input dataset. It is proved theoretically and experimentally [14] that combination of kernels work well when kernels do not share the same features. A comparison of the words common between the different input types is shown in Table 12. 78% of anchor text is present in either extended/web page and 75% of extended text is present in webpage. Even with high common features between them the support vector overlap between these methods is low only 56% leading to good performance of the classifier.
<table>
<thead>
<tr>
<th>Method</th>
<th>Anchor Text</th>
<th>Extended Text</th>
<th>Web Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anchor</td>
<td>300</td>
<td>210</td>
<td>231</td>
</tr>
<tr>
<td>Extended</td>
<td>210</td>
<td>1498</td>
<td>1120</td>
</tr>
<tr>
<td>Web Page</td>
<td>231</td>
<td>1120</td>
<td>3506</td>
</tr>
</tbody>
</table>

Table 12: No. of words common between input types
CHAPTER XI

RESULTS

An experiment was carried out by using the crawler to fetch around 1000 web pages from the web. The crawler was given a Bioinformatics related web page as the starting node.

The SVM was trained with all the data (360 web pages) and with linear Kernel, stop words were removed, stemming, document frequency of 2, C=1, tolerance=0.001 and epsilon = 0.001. Used one-against-all method to multiclass classify the data and the threshold for all the categories was set to 0 as this setting gave good results on the test data. The citation documents were not separately obtained like in training but the documents within the 1000 web pages that had the citation links were used.

Table 13 shows the result from classifying these 1000 web pages. Out of 1000 web pages only 11% of the pages were classified into actual categories, 20% belonged to discard category, 50% were not classified and 19% were duplicates that is they had the same base url in the same category of already categorized urls. Nine pages were classified in more than one category.

Table 14 shows the classification accuracy in each of the categories. All the web pages that were not classified into any category (null) were added to the discard category for performance calculations.

Micro-averaging the results gave a precision and recall value of 87.22%. Macro-averaging the results gave a precision of 50.22% and recall of 52.38%. The categories
company and software had all the classified pages incorrect and actually they did not have even one web page out of the 1000 web pages meant for them. The web pages discarded were highly correct and this was the bulk of the data about 70%. The categories Education and services had good accuracy even though they were rare categories.

Time taken by the classifier to test was very low (hardly a second). Time taken for data converter is much more than the classifier as it has to read many files especially the citing documents to create the bag of words for each data.

The overall result has been good even with such low amount of training data and with citation documents not providing adequate information. Training with more data would definitely help. This can be done as a continuous learning process for the classifier as more and more web pages are added to the world wide web. Certainly some manual intervention is required to analyze the results and the frequency of this analysis can be reduced as the classifier performance increases with training and fine tuning.

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Companies</td>
<td>57</td>
</tr>
<tr>
<td>Education</td>
<td>2</td>
</tr>
<tr>
<td>Research</td>
<td>6</td>
</tr>
<tr>
<td>Services</td>
<td>13</td>
</tr>
<tr>
<td>Software</td>
<td>35</td>
</tr>
<tr>
<td>Discard</td>
<td>206</td>
</tr>
<tr>
<td>Null</td>
<td>495</td>
</tr>
<tr>
<td>Duplicates</td>
<td>191</td>
</tr>
</tbody>
</table>

Table 13: Classification of Test Data
<table>
<thead>
<tr>
<th>Category</th>
<th>Correct Classified</th>
<th>Incorrect Classified</th>
<th>Correctly UnClassified</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Companies</td>
<td>0</td>
<td>57</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Education</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Research</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>Services</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>76.9</td>
<td>76.9</td>
</tr>
<tr>
<td>Software</td>
<td>0</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Discard</td>
<td>697</td>
<td>4</td>
<td>100</td>
<td>99.42</td>
<td>87.42</td>
</tr>
</tbody>
</table>

Table 14: Classification Accuracy of Test Data
CHAPTER XII

PROGRAMS

The following lists the programs that are part of this model. They are listed in the order used when creating, training and testing a technical area. Most of the programs interact with the database.

- ExtractLink.pl: Perl program that can be used to extract links from directory(category) listings from sites like google. It creates a file containing the link along with the specified category.

- fetchURL.pl: Perl Program that fetches the url from the file created by ExtractLink.pl and stores in the given directory and also stores the information in the database.

- Google.pl: Perl program takes the url from the database inserted by fetchURL.pl and fetches citing documents and extracts extended and anchor text from them and stores it in the specified directory.

- DataSplit.pl: Splits the data stored in the database by fetchURL.pl into training and testing based on the mentioned percentage

- InfoGain.pl: Perl program gets the training data information from web documents obtained by fetchURL.pl or google.pl or both and converts it into a word list sorted by information gain value. It can remove words that do not
have the specified document frequency. It removes stop words and stems the words

- loadVector.pl: Perl program that loads the training and testing data (vector format) in the database using the word list (from InfoGain.pl). It can ignore words that do not have the specified information gain value.

- smo: C++ binary which is the classifier. It fetches the data stored in the database and classifies it based on the SVM parameters it's been given like kernel type, multiclassification model, etc. Once it finishes training, it classifies the test data and updates the database with the results.

- PreRecall.pl: Perl program which calculates the precision and recall (both micro-averaging and macro-averaging) for test data results.

- crawler.pl: Perl program that crawls the web for web links. Stores all the data in specified directory and information is stored in the database.

- loadCrawlerData.pl: Perl program that loads the crawler data into the database in vector format.

- updateClassId.pl: Perl program that updates the crawler web link with class id. It eliminates duplicates of base url
CHAPTER XIII

CONCLUSION

Support Vector machines have both theoretical and empirical evidence that they are well suited for text categorization[13]. This approach has been well applied in the area of classifying technical web pages. The input to the classifier is very important and the performance has been increased by using text around links from citation documents along with the web document itself. By various tests it is proved in this thesis that using combination of all the different text has better performance, speed and ease of implementation.

The web portal model has been designed to provide flexibility to add multiple new areas, train, test and measure performance. Since results are stored in database it is much cleaner and easier to analyze. Experiment was conducted on an emerging technical area and even with insufficient amount of training data, the classifier performed well. Manual evaluation can be greatly reduced in classification using this model.

The time required to pre-process the web documents is much more than the time required to classify. Some means of reducing this pre-processing time could be future research. The extended text used words around the link and in future a more sophisticated approach like using structural information of the citing document can be tried. Classification of web pages using support vector machine has been proved possible and can be very effective and can be applied to any Technical field with ease.
BIBLIOGRAPHY


[13] T. Joachims, Text Categorization with support vector machines: Learning with many relevant features


APPENDIX A

USER GUIDE

This guide will demonstrate how to use the web portal model to train, test and display a new topic. The whole process is done in steps. In each step you can use different parameters to get different results.

All users should be familiar with the unix system and have working knowledge on mysql database. Also some perl, php and c++ programming skills will also be helpful to understand the system better.

The code, scripts and documents are all arranged in the following manner under the main directory dira.uccs.edu/~jmuthu/project

- **bin** - consists of all perl scripts and executables
- **code** - consists of the c++ code for the classifier
- **data/doc** has all training and test data
- **data/src** - has files for each category containing the training and test links
- **output** - consists of all outputs from the scripts
- **result** - consists of output from the classifier
- **sql** - consists of database scripts
- **web** - consists of all web pages for the web portal
Database

You need to create a database or you can use an existing database. Nine tables need to be created using the script sql/create.sql

Fetching web pages

For a given new topic for example cancer there will be many categorizes like research, hospital, support etc. Each category may have subcategories. For each of the final subcategories (which are the leaves of this hierarchy tree) there will be a list of web pages. These web links will be used for training and testing the categories (leaves). You can get the initial web pages from web directories like yahoo/google or you can manually create the list. You need to store the web links along with their category name in a file. Then run the command

    bin/fetchURL.pl data_set_name data_directory_name URL_list_file

where
data_set_name is name of the topic,
data_directory_name is the directory in which the web pages are to be stored, URL_list_file is the file which contains the web links along with their category name.

This script should fetch the web pages from the internet and store them in the given directory and update the database table Data.

Fetching anchor/extended text
For fetching citation anchor and extended text for the web links, run the following command

```
bin/google.pl data_set_name starting_data_id
```

where

data_set_name is name of the topic,

starting_data_id is the database id of the web link stored in the table Data from which you want google to start from.

Google.pl fetches 99 web links from the starting data id as google restricts number of queries to 1000 results. For each web link, this program fetches 10 citation links. It extracts the anchor text and extended text from these citations and stores them in the same directory as the web link and also uses the web page filename of the web link. Anchor text is stored with an extension of anc and extended text is stored with an extension of ext.

**Training/Testing the data**

First you may want to split the data into training and testing data. Run the following command

```
bin/dataSplit.pl data_set_name percentage
```

where

data_set_name is name of the topic,

percentage is the percentage of web pages that should be training data.

Next extract the features from the training data using the following command

```
bin/InfoGain.pl data_set_name frequency_limit load_all_data_for_training(y/n) [file_extension]
```

where
data.set.name is name of the topic,
frequency.limit is the minimum limit for no. of occurrences of a word,
load.all.data.for.training(y|n) this argument decides whether you want to load
all data or just training data,
file.extension extension of the input type. Use anc for anchor text and ext for
extended text.

You could create different input types for a web link by combining one or
more of the following - web page, extended text and anchor text. Then store the
concatenated file with an unique extension. Remember different input types differ
only by extensions and the first part of the filename remains the same.

This program outputs all the words extracted along with Information gain value.
You need to store this result in a file as it will be used to convert training/test data
into vector format.

`loadVector.pl word_list_file minimum_IG_score group_name data_set_name
data_type load_all_data_for_training(y|n) [file_extension]`

where word_list_file the output from InfoGain.pl,
minimum_IG_score is the minimum information gain limit for a word,
group_name can be same as topic name or can be a unique name if you have different
input types for a topic,
data_set_name is name of the topic,
data_type is the type of data (sparse, sparsebinary, dense),
load_all_data_for_training(y|n) this argument decides whether you want to load
all data or just training data,
file_extension extension of the input type. Use anc for anchor text and ext for
extended text.

The above program loads the database tables converting the training/test text
into a vector format. Now run the classifier to train/test the data.
bin/smo -c Constant -t tolerance -e epsilon -p two_sigma_squared
-r random_seed -d -l -s -a class_id_against -f class_id_for
-g group_name

If group_name is mentioned then classifier runs all the data for the classes
belonging to the group. If group_name is omitted, it will try to train the data for the
class class_id_for against the class class_id_against. If -l is mentioned then linear
kernel is used and if it is omitted then guassian kernel is used. If -d is mentioned
then DAG multiclassification method is used and if it is ommitted then one against all
multiclassification method is used. Classifier updates the database with the train/test
results.

To check the performance of the trained classifier run the following command

bin/PrecRecall.pl group_name

where group_name is the name of the group.

This ends the training/testing phase. You can repeat the above steps by chang-
ing input parameters to obtain better results. This whole process is like a pipe process
so you will have to follow the above process order.

**Crawling for new links**

Once you finish training and testing the classifier for all the categories, you are
ready to crawl the internet for new links for the topic.

bin/crawler.pl data_directory_Name starting_URL Max_URL_to_be_Fetched

where data_directory_Name is the directory where the fetch web pages are to be
stored,

starting_url is the url that crawler uses as a starting point,

max_url_to_be_fetch is the maximum number of web pages to fetch.
The crawler fetches the web pages and stores them in the directory and updates the database. To classify these pages we need to convert them into vector format using the following command

```
bin/loadCrawlerData.pl group_name skip_citing_links(y|n) ['all']
```

where `group_name` is the name of the group,
`skip_citing_links(y|n)` whether to extract anchor and extended text,
`all` - all data needs to be classified irrespective of whether they have already been classified

This program loads the data into the database so the classifier can classify them. The classifier is run using the smo command mentioned above. After the classifier is finished, we need to update the results so that the web links are displayed under the proper category using the following command.

```
bin/updateClassId.pl group_name
```

where `group_name` is the name of the group.

The web pages will now display the newly classified links. You can modify the web pages to display in the manner you want. These programs can be automated as a cron job to run daily to fetch/classify new links.