A Rough Set-Aided System for Sorting WWW Bookmarks

Richard Jensen

MSc in Artificial Intelligence
Division of Informatics
The University of Edinburgh
2000

Abstract

Most people store 'bookmarks' to web pages. These allow the user to return to a web page later on, without having to remember the exact URI/URL address. People attempt to organise their bookmark databases by filing bookmarks under categories, themselves arranged in a hierarchical fashion. As the maintainence of such large repositories is difficult and time-consuming, a tool that automatically categorises bookmarks is required.

This thesis investigates how well rough set theory (RS) can extract information out of this domain, for use in an experimental automatic bookmark classification system. A comparison is made between this approach to data reduction and a new approach proposed by the author.

The results show that for this domain, RS is successful in reducing datasets whilst retaining their information content. The new approach performs even better than RS for the classification of new bookmarks.

Acknowledgements

I would like to thank my supervisor, Qiang Shen, for helping and guiding me through this project. Additional thanks go to the EPSRC for providing financial support for the year (studentship no: 99407338).

Contents

List of Figures

List of Tables

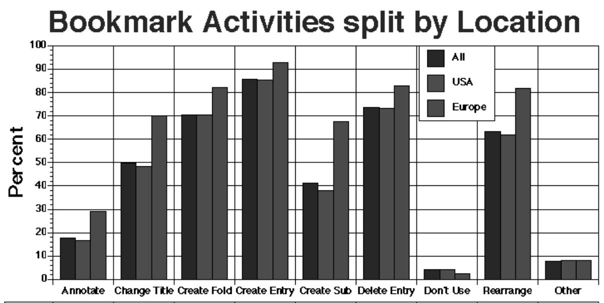
Chapter 1

Introduction

As the use of the Web becomes more prevalent and the size of personal repositories grows, adequately organising and managing bookmarks becomes crucial, somewhat analogous to the need to organise files in a private disk. Several years ago, in recognition of this problem, web browsers included support for tree-like folder structures for organising bookmarks. These enable the user to browse through their repository to find the necessary information. However manual URL classification and organisation can be difficult and tedious when there are more than a few bookmarks to classify - something that goes against the grain of the whole concept of bookmarking.

The purpose of this project is to investigate how well rough set theory can help extract information from a relatively information-poor domain, namely the databases of 'bookmarks' or 'favorites' saved by WWW browsers. This project aims to implement a rough set-aided system that will automatically sort Web bookmarks, to investigate how successful it is and compare its performance with that of an alternative data reduction technique.

Bookmark Activities split by Location



	Annotate	Change Title	Create Fold	Create Entry	Create Sub	Delete Entry	Don't Use	Rearrange	Other
All	17.8	49.67	70.37	85.8	41.32	73.61	4.24	63.19	7.65
USA	16.56	48.37	70.43	85.24	38.17	73.43	4.19	62.04	8.01
Europe	29.18	69.99	82.16	92.96	67.4	82.8	2.35	81.92	7.97

Source: GVU's Eighth WWW User Survey^{tm} (Conducted October 1997)
<URL:http://www.gvu.gatech.edu/user_surveys/>
Copyright 1997 GTRC - ALL RIGHTS RESERVED
Contact: www-survey@cc.gatech.edu

1.1 Bookmarking

An empirical study on users' World Wide Web page revisitation patterns (carried out by (Tauscher and Greenberg 1997)) found that 58% of pages viewed are revisits. So over half of the instances where a user accesses a page, they are revisiting it (probably via their bookmark database). Another survey was carried out by the GVU's WWW Surveying Team (GVU 1997) to determine which bookmarking activities are performed by different people groups. Most respondents create entries (86%), delete entries (74%), create folders (70%) and rearrange entries (63%), with only 4% saying that they don't use them at all. Those creating sub-folders, however, was comparatively low.

This suggests that although people spend time creating and rearranging their bookmarks, the hierarchy tends to have a shallow tree-like structure. This could be for the following reasons:

- Many usability studies, for example (Larson and Czerwinski 1998), indicate that a deep hierarchy results in less efficient information retrieval as many traversal steps are required, so users are more likely to make mistakes.

- Users don't have the time/patience to arrange their collection into a well-ordered hierarchy. Also, if the tree has been ordered and is quite deep, it can take too long to traverse the sub-folders to reach the desired bookmark.

It seems then that there is a need for a tool that can automatically create folders and sub-folders and classify bookmarks into them. The tool should not create deep trees, and so avoid falling prey to the pitfalls outlined above, but should provide quick and easy access to all bookmarks in the database.

1.2 Dimensionality Reduction

The datasets generated in Information Retreival systems tend to be extremely large, rendering most classifiers intractable. This results in the need for a mechanism that will greatly reduce the dimensionality of these datasets, whilst retaining the important information. This project investigates the effectiveness of the rough set approach for this task.

The theory of rough sets was originated by Zdzislaw Pawlak in the late 1970's and is concerned with the classificatory analysis of imprecise, uncertain or incomplete information expressed in terms of data tables (EBRSC 1993). The data can be acquired from measurements or human experts, although it must be discrete.

Rough set theory has already been successfully applied to many areas, including text categorisation (Chouchoulas 1999), but has yet to be applied to the bookmark classification domain. It offers an efficient and useful tool for extracting information from this domain. The technique of Rough Set Data Reduction (RSDR) selects those attributes from a large dataset that contribute most to the classification task at hand, in other words it removes redundancy.

	Microscopic primarily numeric	Macroscopic descriptive and numeric
Deductive	Chaos Theory	Fuzzy methods
Inductive	Neural Networks, genetic algorithms	RSDA

Table 1.1: The position of RSDA in Soft Computing

Rough Set Data Analysis (RSDA) is considered to be part of the "soft computing" paradigm. In (Munakata 1998) it is treated as one of the five key non traditional AI areas (1.1). It differs from the other soft methods in that it does not require additional model assumptions (for example, prior probabilities, fuzzy functions etc), i.e. it only uses the information given by the operationalised data.

1.3 Aims and Objectives

The aim of this project is to investigate how well rough set theory can help extract information from the bookmark domain (a relatively information-poor domain). This has been extended to include the comparison of the rough set approach with an alternative reduction mechanism. This alternative is a new technique developed by the author.

- This project will attempt to apply the rough set concept to the automatic classification of bookmarks.
- It will investigate how well RSDR performs in reducing datasets whilst retaining their information content.
- It will compare the RSDR technique with that of a completely new data reduction technique implemented by the author.

- The design of the system will be modular, allowing components to be replaced with an alternative implementation.
- Various classification techniques will be implemented and experimentation carried out.
- This approach will be analysed and compared with other similar systems.

1.4 Structure

Chapter 2 provides some background to the subjects covered in this project. A brief summary of text classification techniques is given, as well as an overview of Rough Set Theory and ID3/Entropy. Finally, existing systems that are related to this topic are examined and evaluated.

Chapter 3 discusses the theoretical aspects involved. The design of the system is outlined and additional information on the sub-systems is presented. Other issues related to the theory of the project are included.

Based on the previous chapter, this thesis goes on to discuss the implementation of the software and how the theoretical issues influence this in chapter 4. The choice of language used is defended, and the implementation details of each module is given.

Chapter 5 then presents the results obtained from experimentation, and analyses the difference in performance when different techniques are used both in data reduction and in classification. This system is then compared and contrasted with existing systems. This chapter ends with a sample run of the program.

Finally, conclusions are made in chapter 6. The success of the system is assessed, with future improvements and extensions proposed.

Chapter 2

Background

This project employs strategies from various fields to achieve its goals. The main techniques involved are from the Text Categorisation and Data Reduction areas. These are discussed in detail in this chapter.

The automated categorisation of text has been the subject of a large amount of research over the years, dating back to the early 1960s. It is used in many contexts, such as automatic document indexing, automated metdata generation and word sense disambiguation.

Data Reduction is a topic generating a great deal of interest nowadays, particularly with the explosive growth of the internet and accordingly, the amount of available information.

2.1 Text Categorisation

Text categorisation (TC) has often been defined as the content-based assignment of one or more predefined categories to text. As Moulinier (Moulinier 1996) states, it has become important from two points of view. Firstly, it is important from the Information Retrieval (IR) viewpoint because of the rapid growth of textual information sources. These require a greater amount of information processing, so TC can be used to aid IR for this task. Secondly, it is important from the Machine Learning viewpoint (ML) as text categorisation provides ML with an application field. The approach that ML takes in automatic classification is to generate a means of classification by the use of induction over examples that have been categorised previously (a form of supervised learning).

The classification of textual documents involves two main phases: training and classification.The training phase involves examining the document and retrieving those keywords deemed important. These sets of keywords are large, rendering most text classifiers intractable, so a dimensionality reduction step is performed. Induction is carried out, and a means of classifying future data is the output. The classification phase uses the classification means from the training process to classify new documents.

There are many text classification strategies, the most common ones being rule-based, vector-based and probabilistic techniques.

2.1.1 Rule-Based

For rule-based approaches to classification, a set of rules and an appropriate classifier are required. Each individual rule has a set of preconditions and an associated decision. If a document matches the preconditions, then it is classified according to the decision value.

A simple form of this is the *Boolean Exact Model* which employs exact boolean existential rules. This project uses a variant of this called the *Boolean Inexact Model* (described in the next chapter). A more complex approach is the *Fuzzy Rule-Based* technique which uses fuzzy production rules and fuzzy reasoning to classify documents.

2.1.2 Vector-Based

The Vector Space Model (VSM) considers document representatives as binary vectors embedded in an n-dimensional Euclidean space (n is the total number of keywords). As there tends to be a large number of keywords involved, the dimensionality is also very high.

Each document and query is represented as a point in the space. To obtain the document vector, the keywords contained in the document are obtained and ranked according to their weight in the document. They are then converted to a vector by ordering them in some fixed way. Any missing keywords (according to the universal keyword set) are marked as absent. The similarity between a query vector and a document term vector can then be computed as the scalar product of the two.

There are a number of disadvantages with the VSM. There is the lack of justification for some of the vector operations (for example, the choice of similarity function and the choice of term weights). It is barely a retrieval model as it doesn't explicitly model relevance. There is also the assumption that a query and a document can be treated the same.

However, the simplicity of the model is attractive - probably why it is the most popular retrieval model today. It can also measure similarities between almost anything (e.g. documents and queries, documents and documents, etc).

2.1.3 Probabilistic

Another classic retrieval and classification method is the probabilistic retrieval technique (Fuhr 1992), where the probability that a specific document will be judged relevant to a specific query, is based on the assumption that the terms are distributed differently in relevant and non relevant documents. The probability formula is usually derived from Bayes' theorem. Given a particular document x and a set of categories, the probability of x belonging to each category in the category set is calculated. The document is classified into the category that produced the highest probability. The basic model is illustrated below (taken from (van Rijsbergen 1979)):

Any document x can be represented as a binary vector as they are assumed to be described by the presence/absence of index terms.

$x = (x_1, x_2, ..., x_n)$

Each x_n can be 0 or 1 depending on whether the ith index term is present. An additional assumption is that w_1 and w_2 are mutually exclusive events, such that:

$$\begin{aligned} w_1 &= \text{document is relevant} \\ w_2 &= \text{document is non-relevant} \end{aligned}$$

We want to calculate $P(w_1|x)$ and $P(w_2|x)$ for each document to decide which is relevant and which is not. To estimate these we can use Bayes' Theorem, that tells us for discrete distributions:

$$P(w_i|x) = \frac{P(x|w_i)P(w_i)}{P(x)} \qquad i = 1, 2 \tag{2.1}$$

Here $P(w_1)$ is the prior probability of relevance, $P(w_2)$ the prior probability of non-relevance. $P(x|w_i)$ is proportional to the likelihood of relevance or non-relevance given x. From this we can say that if $P(w_1)$ ¿ $P(w_2)$, then the document is relevant. This is summed up in the decision rule:

$P(w_1|x) > P(w_2|x) \Rightarrow x$ is relevant, x is non-relevant D1

which can be rewritten (using Bayes' Theorem) as:

$$[P(x|w_1)\ P(w_1) > P(x|w_2)\ P(w_2) \Rightarrow x \text{ is relevant, } x \text{ is non-relevant}]\ \text{D1*}$$

The basis for this rule is simply that it minimises the average probability of error. Note that for any document x the probability of error is:

$$P(error|x) = \begin{cases} P(w_1|x) & if\ we\ decide\ w_2 \\ P(w_2|x) & if\ we\ decide\ w_1 \end{cases}$$

(2.2)

This is not the only factor worth minimising. We can assign each type of error an associated cost in order to provide a means by which we can minimise *risk*. We define a cost function l_{i_j}, which is the loss incurred for deciding w_i when w_j is the case. The conditional risk (the expected loss when deciding w_i) can now be defined as:

$$R(w_i|x) \ = \ l_{i_1}P(w_1|x) \ + \ l_{i_2}P(w_2|x) \quad i = 1,2 \tag{2.3}$$

this results in the decision rule:

interpreted[$R(w_1|x) < R(w_2|x) \Rightarrow x$ is relevant, x is non-relevant] D2

D2 and D1* can be combined, resulting in the final decision rule:

$$[R(w_1|x) < R(w_2|x)] \Leftrightarrow [(l_{2_1} - l_{1_1})\ P(x|w_1)\ P(w_1) > (l_{1_2} - l_{2_2})\ P(x|w_2)\ P(w_2)]$$

The relevance or non-relevance of document x can now be decided. Given a category set, this process can decide which category document x belongs to.

2.1.4 Latent Semantic Indexing

A brief mention of Latent Semantic Indexing (LSI) is given here. LSI is a variant of the vector retrieval model outlined previously which takes into account the dependencies between terms (Dumais 1996; Deerwester et al 1990). Unlike other models, LSI treats words as if they are not independent of each other; it attempts to automatically derive and model inter-relationships between them.

The term-document matrix can be considered to be a “bag of documents” and is split into a set of k orthogonal factors. Similarity is computed in the following way (from (Oard 1999)):

1. Choose k. (not greater than the number of terms or documents)
2. Add the weighted vectors for each term - multiply each vector by term weight - sum each element separately
3. Repeat for query or second document
4. Compute inner product - Multiply corresponding elements and add.

An advantage that LSI has is that it reduces the original number of dimensions. Vectors that are similar are assigned to similar terms. This composite term is then mapped to a single dimension. However, as with most retrieval models, words with similar meanings confound LSI. Also, the computations required are expensive (but a lot f computation is carried out in advance).

2.1.5 Dimensionality Reduction

In text categorisation, the high dimensionality of the term space can be problematic, so some form of dimensionality reduction is employed. This can also be beneficial as it tends to reduce $overfitting$, where a classifier is tuned also to the contingent, rather than just the necessary characteristics of the training data (Sebastiani 1999). The classic symptom of this is displayed when classifiers that are very good at classifying their training examples are notably worse at classifying new data.

Dimensionality reduction by term selection is the approach adopted in both of the reduction implementations. The techniques attempt to select a subset r' from the original set of r terms such that the reduction in effectiveness is minimal when compared to the original representations. A number of techniques exist:

- *Document Frequency.* This is a simple and effective reduction strategy [Apté et al 1994]. It has been shown that the most informative terms are those with low to medium document frequency. By removing those attributes that occur the most (and to some extent those that occur rarely), the dimensionality of the document is reduced with no or little loss in information. To make this effective stop words should be removed beforehand, otherwise only topic-neutral words may remain after reduction.

- *Stop Word Removal.* Again, this is a simple technique for removing very information-poor terms. Stop words are connectives such as articles and contribute very little (if anything) to the classification of documents. Care must be taken when adopting this approach as it is feasible that some information-rich words might be mis- as stop words.

- *Word Stemming.* Word suffixes are removed, leaving only the root of the word. This is an attempt to reduce words with similar meanings to the same root, for example *retailing* and *retailer* both contain the same root, namely *retail*. This is not guaranteed to work, however, as many words with different meanings share a common root.

- *Other Term Selection Functions.* There has been much research into using sophisticated information-theoretic term selection functions, such as *chi−square* (Shütze et al. 1995), *correlation coefficient* (Ng et al. 1997) and *relevancy score* [Wiener et al. 1995]. These functions have been shown to produce better results than document frequency.

Of course, there are many more techniques in this area. A fuller discussion of these can be found in (Sebastiani 1999). Relating this to the bookmark domain, some strategies are more appropriate than others. Word stemming is not applicable when considering URLs as they contain very few recognisable words (such as "http"). The words that do exist in the URL tend to be the root words anyway. Removal of stop words can be applied to the title part of a bookmark as these contain more natural language style text.

The main dimensionality reduction methods are introduced in the following sections.

2.2 Rough Sets

Rough set theory is a tool for studying imprecision, vagueness, and uncertainty in data analysis. Zdzislaw Pawlak (Pawlak 1982) originated the theory in the 1970s - the end result of a long term program of fundamental research on logical properties of information systems carried out by him and a group of logicians from the Polish Academy of Sciences and the University of Warsaw, Poland.

The rough set itself is the approximation of a vague concept (set) by a pair of precise concepts, called lower and upper approximations (which are a classification of the domain of interest into disjoint categories). The classification (attributes) formally represents our knowledge about the domain. Objects belonging to the same category (same attributes) are not distinguishable.

In this project, rough set theory is used as a tool for data reduction. It is used to discover data dependencies and reduce the number of attributes by purely structural methods. The following example (taken from (Slowinski 1992) and (Pawlak 1991)) should clarify the main concepts involved.

2.2.1 Theory

A dataset can be viewed in the form of a table, where columns are labelled by attributes and rows are labelled by objects. Let **U** be the universe (the set of all objects in the dataset), let **A** denote the set of all attributes in the dataset, **C** the set of condition attributes and **D** the set of decision attributes, so that $\mathbf{C} \subset \mathbf{A}$, $\mathbf{C} \subset \mathbf{A}$, $\mathbf{C} \cup \mathbf{D} = \mathbf{A}$, and $\mathbf{C} \cap \mathbf{D} = \emptyset$. The entry in column q and row x has the value *f(x,q)*, so *f(x,q)* defines an equivalence relation over **U**. Given q, the universe can be partitioned into a set of disjoint subsets:

$$R_q = \{x \,:\, x \epsilon\, \mathbf{U} \wedge f(x,q) = f(x_0,q)\ \forall x_0 \epsilon\, \mathbf{U}\} \tag{2.4}$$

$x\epsilon\mathbf{U}$	a	b	c	d	$\Rightarrow$	e
0	1	0	2	2		0
1	0	1	1	1		2
2	2	0	0	1		1
3	1	1	0	2		2
4	1	0	2	0		1
5	2	2	0	1		1
6	2	1	1	1		2
7	0	1	1	0		1

Table 2.1: An example dataset

Using the example dataset in table (2.1), we have $\mathbf{U} = \{0,1,2,3,4,5,6,7\}$, $\mathbf{A} = \{a,b,c,d,e\}$, $\mathbf{C} = \{a,b,c,d\}$, $\mathbf{D} = \{e\}$, resulting in the following partitions:

$$R_a = \{\{1,7\},\{0,3,4\},\{2,5,6\}\}$$
$$R_b = \{\{0,2,4\},\{1,3,6,7\},\{5\}\}$$
$$R_c = \{\{2,3,5\},\{1,6,7\},\{0,4\}\}$$

$$R_d = \{\{4,7\},\{1,2,5,6\},\{0,3\}\}$$
$$R_e = \{\{0\},\{2,4,5,7\},\{1,3,6\}\}$$

2.2.2 Indiscernibility

Central to Rough Set theory is the concept of discernibility. For two objects with different decision attribute values, we would like to be able to discern between them, based on the values of the condition attributes. Let $\mathbf{P} \subset \mathbf{A}$, then two objects *x,y* ϵ $\mathbf{U}$ are indiscernible by the set of attributes $\mathbf{P}$ in our table if and only if *f(x,q)* = *f(y,q)* $\forall$ q ϵ $\mathbf{P}$. For every $\mathbf{P} \subset \mathbf{A}$ we have an indiscernibility relation *IND(P)*. The equivalence classes, or partition of $\mathbf{U}$, generated by *IND(P)* is denoted $\mathbf{U}$/*IND(P)* and can be calculated as follows:

$$\mathbf{U}/IND(P) = \otimes\{q\epsilon\mathbf{P} : \mathbf{U}/IND(q)\}, where \tag{2.5}$$

$$A \otimes B = \{X \cap Y : \forall X\epsilon A, \forall Y\epsilon B, X \cap Y \neq \emptyset\} \tag{2.6}$$

If $P = \{b,c\}$, then objects 1, 6 and 7 are indiscernible; as are objects 0 and 4. *IND(P)* creates the following partition of $\mathbf{U}$:

$$\begin{aligned}
\mathbf{U}/IND(P) &= \mathbf{U}/IND(b) \otimes \mathbf{U}/IND(c)\\
&= \{\{0,2,4\},\{1,3,6,7\},\{5\}\} \otimes \{\{2,3,5\},\{1,6,7\},\{0,4\}\}\\
&= \{\{0,2,4\} \cap \{2,3,5\}, \{0,2,4\} \cap \{1,6,7\}, \{0,2,4\} \cap \{0,4\},\\
&\quad \{1,3,6,7\} \cap \{2,3,5\}, \{1,3,6,7\} \cap \{1,6,7\}, \{1,3,6,7\} \cap \{0,4\}\\
&\quad \{5\} \cap \{2,3,5\}, \{5\} \cap \{1,6,7\}, \{5\} \cap \{0,4\}\}\\
&= \{\{2\},\{0,4\},\{3\},\{1,6,7\},\{5\}\}
\end{aligned}$$

2.2.3 Approximation

As mentioned earlier, a rough set is a set defined by its upper and lower approximations. Given a set $P \subseteq \mathbf{U}$, these are defined as:

$$\underline{P} = \bigcup\{X : X \in \mathbf{U}/IND(P), X \subseteq Y\} \tag{2.7}$$

$$\overline{P} = \bigcup\{X : X \in \mathbf{U}/IND(P), X \cap Y \neq \emptyset\} \tag{2.8}$$

For example if $Y = \{2,4,5,7\}$ and $P = \{b,c\}$ then

$$\underline{P} = \{2, 5\}$$
$$\overline{P} = \{0, 1, 2, 4, 5, 6, 7\}$$

Let P and Q be equivalence relations over $\mathbf{U}$, then the positive, negative and boundary regions can be defined as:

$$POS_P(Q) = \bigcup_{X \in Q} \underline{P}X$$
$$NEG_P(Q) = \mathbf{U} - \bigcup_{X \in Q} \overline{P}X$$
$$BND_P(Q) = \bigcup_{X \in Q} \overline{P}X - \bigcup_{X \in Q} \underline{P}X$$

The positive region, $POS_P(Q)$, contains all objects of $\mathbf{U}$ that can be classified to classes of $\mathbf{U}/Q$ using the knowledge in attributes P. The boundary region, $BND_P(Q)$, is the set of objects that can possibly, but not certainly, be classified in this way. The negative region, $NEG_P(Q)$, is the set of objects that cannot be classified to classes of $\mathbf{U}/Q$.

For example, let $P = \{b,c\}$ and $Q = \{e\}$, then

$$POS_{IND(P)}(IND(Q)) = \bigcup\{\emptyset, \{2, 5\}, \{3\}\} = \{2, 3, 5\}$$
$$NEG_{IND(P)}(IND(Q)) = \mathbf{U} - \bigcup\{\{0, 4\}, \{2, 0, 4, 1, 6, 7, 5\}, \{3, 1, 6, 7\}\} = \emptyset$$
$$BND_{IND(P)}(IND(Q)) = \mathbf{U} - \{2, 3, 5\} = \{0, 1, 4, 6, 7\}$$

This means that objects 2, 3 and 5 can certainly be classified as belonging to a class in attribute e, when considering attributes b and c. The rest of the objects cannot be classified as the information that would make them discernible is absent.

2.2.4 Dependency Discovery

An important issue in data analysis is discovering dependencies between attributes. Intuitively, a set of attributes Q depends totally on a set of attributes P, denoted $P \Rightarrow Q$, if all attribute values from Q are uniquely determined by values of attributes from P. If there exists a functional dependency between values of Q and P, then Q depends totally on P. Dependency can be defined in the following way:

For $P,Q \subset \mathbf{A}$, we say that Q depends on P in a degree k ($0 \leq k \leq 1$), denoted $P \Rightarrow_k Q$, if

$$k = \gamma_P(Q) = \frac{|POS_P(Q)|}{|\mathbf{U}|} \tag{2.9}$$

If $k = 1$ we say that Q depends totally on P, and if k ¡ 1 we say that Q depends partially (in a degree k) on P. For example, the degree of dependency of attribute $\{e\}$ from the attributes $\{b,c\}$ is:

$$\begin{aligned}\gamma_{\{b,c\}}(\{e\}) &= \tfrac{|POS_{\{b,c\}}(\{e\})|}{|\mathbf{U}|} \\ &= \tfrac{|\{2,3,5\}|}{|\{0,1,2,3,4,5,6,7\}|} \\ &= \tfrac{3}{8} = 0.375\end{aligned}$$

Thus $P \Rightarrow_{0.375} Q$. The complement of γ gives a measure of the contradictions in the knowledge. In the example, the measure of contradiction is 0.625, so five objects represent contradictions whilst the remaining three can be classified into the decision attribute e.

2.2.5 Attribute Significance

By calculating the change in dependency when an attribute is removed from the set of considered conditional attributes, we can get a measure of the significance of the attribute. The higher the change in dependency, the more significant the attribute is. If the significance is 0, then the attribute is dispensible. More formally, given P, Q and an attribute $x \epsilon P$,

$$\sigma_P(Q, x) = \gamma_P(Q) - \gamma_{P-\{x\}}(Q) \tag{2.10}$$

For example, if $P = \{a,b,c\}$ and $Q = e$ then

$$\begin{aligned}
\gamma_{\{a,b,c\}}(\{e\}) &= |\{2,3,5,6\}|/8 = 4/8 \\
\gamma_{\{a,b\}}(\{e\}) &= |\{2,3,5,6\}|/8 = 4/8 \\
\gamma_{\{b,c\}}(\{e\}) &= |\{2,3,5\}|/8 = 3/8 \\
\gamma_{\{a,c\}}(\{e\}) &= |\{2,3,5,6\}|/8 = 4/8
\end{aligned}$$

And calculating the significance of the three attributes gives:

$$\begin{aligned}
\sigma_P(Q, a) &= \gamma_{\{a,b,c\}}(\{e\}) - \gamma_{\{b,c\}}(\{e\}) = 1/8 \\
\sigma_P(Q, b) &= \gamma_{\{a,b,c\}}(\{e\}) - \gamma_{\{a,c\}}(\{e\}) = 0 \\
\sigma_P(Q, c) &= \gamma_{\{a,b,c\}}(\{e\}) - \gamma_{\{a,b\}}(\{e\}) = 0
\end{aligned}$$

From this we can conclude that attribute a is indispensible, but attributes b and c can be dispensed with.

2.2.6 Attribute Reduction

The reduction of attributes is achieved by comparing equivalence relations generated by sets of attributes. Attributes are removed so that the reduced set provides the same quality of classification as the original. A *reduct* is defined as a subset R of the conditional attribute set $\mathbf{C}$ such that $\gamma_R(\mathbf{D}) = \gamma_C(\mathbf{D})$. A given dataset may have many attribute reduct sets, so the set R of all reducts is defined as:

$$\mathsf{R} = \{X : X \subseteq \mathbf{C}, \gamma_X(\mathbf{D}) = \gamma_C(\mathbf{D})\} \tag{2.11}$$

The intersection of all the sets in R is called the *core*, the elements of which are those attributes that cannot be eliminated without introducing more contradictions to the dataset. In Rough Set Dimensionality Reduction (RSDR), a reduct with minimum cardinality is searched for, in other words an attempt is made to locate a single element of the minimal reduct set $\mathsf{R}_{min} \subseteq \mathsf{R}$:

$$\mathsf{R}_{\mathsf{min}} = \{X : X \epsilon\, \mathsf{R},\ \forall Y \epsilon\, \mathsf{R}, |X| \leq |Y|\} \tag{2.12}$$

Using the example, the dependencies for all possible subsets of $\mathbf{C}$ can be calculated:

$$\begin{array}{cc}
\gamma_{\{a,b,c,d\}}(\{e\}) = 8/8 & \gamma_{\{b,c\}}(\{e\}) = 3/8 \\
\gamma_{\{a,b,c\}}(\{e\}) = 4/8 & \gamma_{\{b,d\}}(\{e\}) = 8/8 \\
\gamma_{\{a,b,d\}}(\{e\}) = 8/8 & \gamma_{\{c,d\}}(\{e\}) = 8/8 \\
\gamma_{\{a,c,d\}}(\{e\}) = 8/8 & \gamma_{\{a\}}(\{e\}) = 0/8 \\
\gamma_{\{b,c,d\}}(\{e\}) = 8/8 & \gamma_{\{b\}}(\{e\}) = 1/8 \\
\gamma_{\{a,b\}}(\{e\}) = 4/8 & \gamma_{\{c\}}(\{e\}) = 0/8 \\
\gamma_{\{a,c\}}(\{e\}) = 4/8 & \gamma_{\{d\}}(\{e\}) = 2/8 \\
\multicolumn{2}{c}{\gamma_{\{a,d\}}(\{e\}) = 3/8}
\end{array}$$

The dataset is consistent since $\gamma_{\{a,b,c,d\}}(\{e\}) = 1$. The reduct and minimal reduct sets for this example are:

$$\mathsf{R} = \{a,b,d\}, \{a,c,d\}, \{b,c,d\}, \{b,d\}, \{c,d\}$$
$$\mathsf{R}_{min} = \{\{b,d\}, \{c,d\}\}$$

If $\{b,d\}$ is chosen, then the dataset can be reduced accordingly:

A method for finding a reduct of minimal cardinality is available and is discussed in chapter 3.

$x\epsilon\mathbf{U}$	b	d	$\Rightarrow$	e
0	0	2		0
1	1	1		2
2	0	1		1
3	1	2		2
4	0	0		1
5	2	1		1
6	1	1		2
7	1	0		1

Table 2.2: Reduced dataset

2.3 ID3 and Entropy

Developed by Quinlan in 1979, the Iterative Dichotomiser version 3 (Quinlan 1986) is a decision-tree building algorithm which adopts a divide-and-conquer strategy for object classification. Originally it divided examples into two classes only (hence the name), but many modern implementations now operate with many-valued classes. However, it is not the algorithm that is of interest here, instead it is the heuristic that ID3 employs for determining which attribute provides the most gain in information.

The entropy of attribute A (which can take values $a_1 ... a_m$) with respect to the conclusion C (values $c_1 ... c_n$) is defined as:

$$E(A) = -\sum_{j=1}^{m} p(a_j) \sum_{i=1}^{n} p(c_i|a_j)\ log_2\ p(c_i|a_j)$$

The attribute with the lowest entropy is the one that has the highest information gain, and so is the most useful determiner. An example will help clarify the process (from (Baltzersen 1995)).

Day of week	Time of day	Weather	Sickness	Mood
Monday	morning	rainy	no	good
Wednesday	evening	rainy	no	bad
Saturday	evening	snowy	no	good
Sunday	morning	rainy	no	bad
Sunday	morning	sunny	no	good
Wednesday	afternoon	sunny	yes	bad
Saturday	evening	snowy	yes	bad
Wednesday	evening	sunny	yes	bad

Table 2.3: Example Dataset

The attributes here are *day, time, weather,* and *sickness*, with *mood* as the decision attribute. Using the entropy function defined above we can calculate the entropy for attribute *sickness*:

$$p(good|no) = 3/5$$
$$p(bad|no) = 2/5$$
$$p(no) = 5/8$$
$$p(good|yes) = 0/3$$
$$p(bad|yes) = 3/3$$
$$p(yes) = 3/8$$

$$\begin{aligned} E(sickness) &= 5/8(3/5log(3/5) + 2/5log(2/5)) \\ &+ 3/8(0/3log(0/3) + 3/3log(3/3)) \\ &= 0.60684 \end{aligned}$$

Repeating this process for the remaining attributes gives:

$E(day) = 0.5$
$E(time) = 0.75$
$E(weather) = 0.93872$

The attribute selected is *day* as it has the lowest entropy. The set of examples is then partitioned according to the possible values for *day*. This process is repeated on each subset. In this example we discover that attribute *time* is irrelevant. In rough set terms this gives us the reduct $\{day, weather, sickness, mood\}$.

Using this process, it is possible to generate an alternative data reduct for datasets. By limiting the number of attributes the algorithm returns, a reduced set of the "best" attributes can be obtained. The original dataset can now be reduced by removing those attributes not present in the reduced dataset.

It is a variant of this technique called Entropy-Based Reduction that is implemented and investigated in this project and compared with RSDR.

2.4 Existing Systems

As this area of research is relatively new, there are very few papers on this topic. Most bookmark utilities provide no means of automatic classification; the systems outlined here are the only ones with such functionality.

2.4.1 Bookmark Organiser

Bookmark Organiser (BO) by (Maarek and Shaul 1996) , is an application that attempts to provide automatic assistance in organising bookmark repositories by their conceptual categories. It can operate in two modes:

1. **Fully Automatic**: If the user does not know which category the bookmark belongs to, they can request BO to insert it in the relevant folder by applying an automatic clustering technique to the document to which it refers.

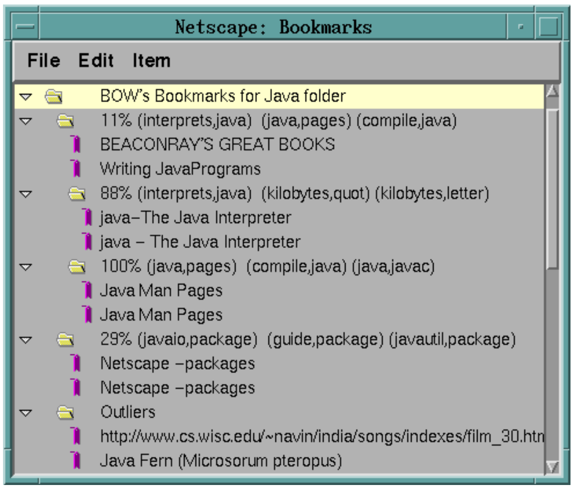
2. **Semi Automatic**: In this case, the user specifies the node into which to insert the new bookmark. They can now request the bookmarks to be re-organised automatically.

Bookmarks are organised by applying sophisticated clustering techniques to the text contained in the referred-to document. The clustering method most used in IR applications (and used for the BO application) is the Hierarchical Agglomerative Clustering (HAC) technique. This is outlined below:

- **Start** with a set of singleton clusters, each contains one object
- **Repeat** the following steps iteratively **until** there is only one cluster
    - **Identify** the two clusters that are the most similar
    - **Merge** them together into a single cluster

This system is a reasonable attempt at automatic categorisation; it constructs folders based on the text in the document (which should give a clear indication as to the category to which it belongs), and also enables automatic and semi-automatic classification (a useful feature).

However, it has many limitations. As seen from the screenshot of the bookmarks database after BO has executed (Figure 2.1), the automatically-generated folder titles are quite unclear and often meaningless to the user. Another significant limitation is that classification accuracy is highly dependent on the number of documents. Running BO on a small number of bookmarks will generate folders containing loosely-related documents, so a large bookmark database is required to achieve any useful results. This links with the next point - there is no indication of the time it takes BO to run through the database and categorise accordingly. The program must examine those documents that each URL refers to and execute its IR process on



BOW's Bookmarks for Java folder
11% (interprets,java) (java,pages) (compile,java)
BEACONRAY'S GREAT BOOKS
Writing JavaPrograms
88% (interprets,java) (kilobytes,quot) (kilobytes,letter)
java–The Java Interpreter
java – The Java Interpreter
100% (java,pages) (compile,java) (java,javac)
Java Man Pages
Java Man Pages
29% (javaio,package) (guide,package) (javautil,package)
Netscape –packages
Netscape –packages
Outliers
http://www.cs.wisc.edu/~navin/india/songs/indexes/film_30.ht
Java Fern (Microsorum pteropus)

Figure 2.1: A BO-categorised bookmark hierarchy

the document text. For large databases this could be too slow for the patience of any user, but as stated previously, large databases are required if satisfactory categories are to be constructed.

2.4.2 PowerBookmarks

This semi-structured database application (developed by (Li et al 1999)) aims to provide personalised organisation and management of bookmarks in a multi-user environment.

PowerBookmarks automatically classifies documents by contents; it parses metadata from bookmarked URLs to index and classify them. It avoids the verbose labelling problem encountered in BO by using existing classifiers with manually selected labels (for example "Sports/Football"). Significant keywords are extracted from documents according to the word frequency analysis. Queries are then issued to an external classifier, the results of which are processed by a classification tree maintainence algorithm to stabilise the tree structure.

The idea of sharing bookmarks is a good one. The bookmark collections

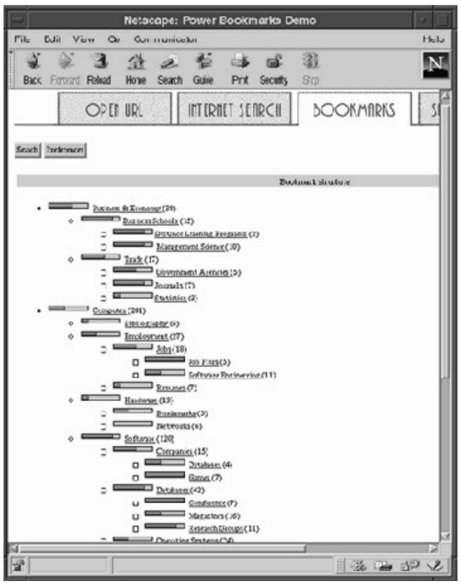


Figure 2.2: PowerBookmark hierarchy

of other people provide a potentially valuable information resource. Instead of using a search engine to locate information, a user can browse the hierarchy to locate any relevant documents. Having bookmarks on the web means that they can be accessed from anywhere in the world, not just at the users' home terminal. The categorisation process used is quite accurate as it uses the text contained in the document. The process also results in clear and concise folder titles.

As this is designed for a shared, web-based environment, it is not particularly useful for the ordinary user who does not have permanent internet access. Every time they wish to browse through their bookmark collection they have to be connected. However, the ideas and strategies implemented are quite successful.

2.5 Related Work

2.5.1 Email Classification

This project is based on research carried out previously by Chouchoulas (Chouchoulas 1999) . He investigates the applicability of rough set theory to the information retrieval and filtering domain, and then exemplifies this by means of an email categorisation application.

The implemented system is modular to address the problem of dimensionality (a typical text categorisation problem). During the training phase, the system acquires keywords from a subset of the total collated email messages and reduces this dataset using rough set theory. This reduced dataset is used as a set of rules by the classifier module for categorising new messages in the testing phase.

The results of this research show that rough set theory is very much applicable to the domain of text categorisation. A classification accuracy of 90% - 100% was obtained for the email categorisation application, showing that precision in classification is only slightly reduced.

The design of this project is based on his system, using a modular approach to allow alternative modules to be tested. However, no comparison of alternative reduction techniques was attempted - an interesting topic worthy of much more research. It is well known that rough sets is a useful tool for data reduction, but how does it compare with other techniques? This project has been extended to address this question.

2.5.2 Web Page Classification

Very recently, research has been carried out into the area of Web page classification. In fact, it has given rise to specific techniques for problems such as indexing, term selection etc as Web pages can be considered to be a special kind of document. Web pages contain text much like any other document, but the fact that they contain pointers to *other* documents makes them an interesting area for research.

An indexing technique specific to these documents has been proposed by (Atardi et al. 1999). A Web page tends to have many other pages pointing towards it. Atardi et al. reason that these documents can be combined forming an artificial document which can be considered to be a compilation of "short reviews" of the Web page. It is this compilation that is used in classification, not the original document.

Another indexing technique has been put forward by (Fuhr et al 1998), where a document representation is obtained by the combined indexing of both the original document and its children (the documents that it points to). This is of interest for Web site classification, as often the children of a document contain relevant information for the site as a whole.

The topic of indexing is not the only one to have been investigated - different methods of classifier induction have been proposed. Chakrabarti et al. (Chakrabarti et al 1998) base their Web page categorisation approach on the hypothesis that for each document-classification pair (d, c), two values can be associated; the authority $a(d, c)$ and the hub value $h(d, c)$. The authority value is a measure of the "authoritativeness" of d on c in terms of the number of c-related pages pointing to it. The hub value measures the "informativeness" of d on c in terms of how many c-related documents it points *to*. Therefore, the purpose of this system is to identify the n most authoritative and the n most informative documents that are associated with the document, given a classification c. By issuing a query c to AltaVista, an initial set of relevant documents is obtained, giving a starting point for the induction algorithm.

Ruiz and Srinivasan (Ruiz and Srinivasan 1999) have developed a method for the induction of classifiers for hierarchical category sets, using neural networks to achieve this.

Web page classification is a closely-related topic to bookmark classification. Both are concerned with classifying web pages, but using quite different information for the task. It is possible that areas of research involved in Web page classification can be modified for the bookmark domain, such as examining the links that a document contains. The URLs and titles of these documents could be combined with those belonging to the original document for a more accurate classification.

2.6 Summary

This chapter has covered the main areas involved in the bookmark classification process. Several techniques for classification have been outlined for this task. For these techniques to be usable, a data reduction step must be taken beforehand. The two proposed methods to achieve this are Rough Set Data Reduction and a new technique called Entropy-Based Reduction. Both of these can be used to reduce any dataset, but are thought to be pontentially very useful for this particular domain.

Related systems have been examined, although there are only a few as this topic is relatively new. This project intends to build on the advantages and disadvantages present in these systems.

Chapter 3

Theoretical Aspects

As mentioned in the previous chapter, the application of rough sets to the text classification domain has been successful, but has never been applied to the domain of bookmark classification. Bookmark databases are a relatively information-poor domain, however, so steps must be taken to ensure that all relevant information is used in the classification process, with any misleading or useless data removed.

This chapter addresses the issues involved in constructing a system that will classify bookmarks. Some of the theory involved in the reduction of data has already been discussed in chapter 2; their function is described here in more detail.

3.1 Design

The system is modular, to allow various components to be replaced with other components to compare and contrast their operation. It may well be the case that rough set theory is not the best way to reduce the datasets involved, so an alternative reduction technique can replace it. In the future, there may be a quicker way to discover a minimal reduct for a dataset than the QuickReduct algorithm - this can then be placed into the system with

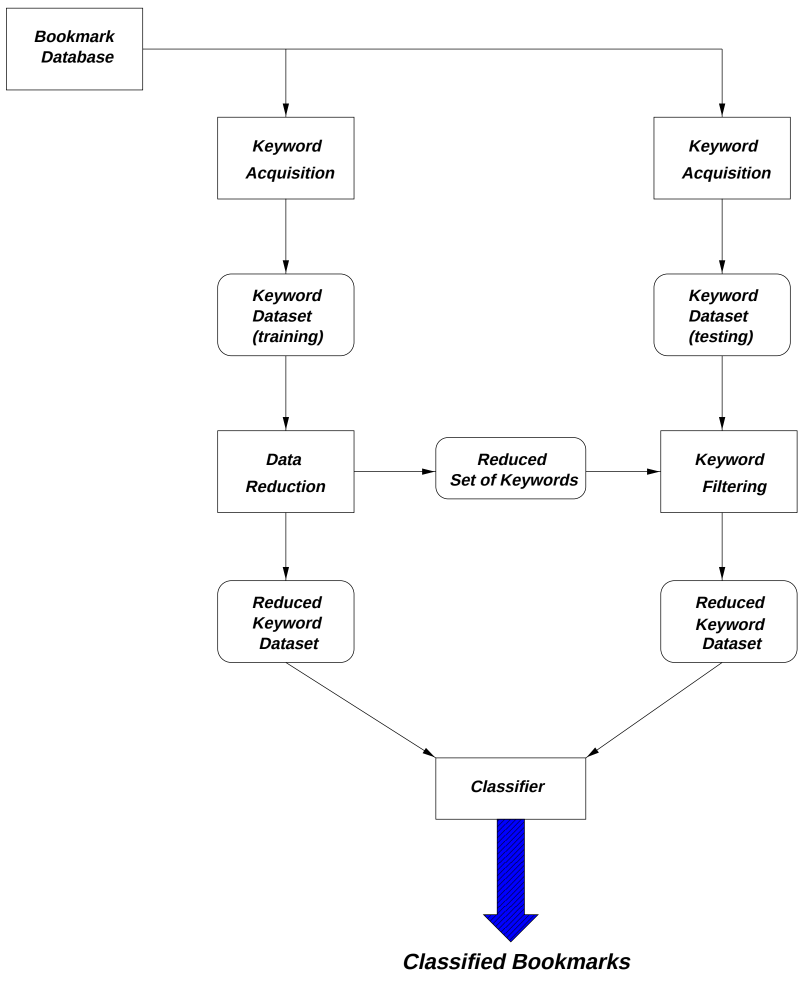


Figure 3.1: The Proposed System

almost no changes to existing parts required.

- *Training and Testing Split* A large dataset of bookmarks was collected from various sources and used in the training phase, while other smaller datasets were used for testing. The format of both datasets is identical.

- *Keyword Acquisition* Given the input from the previous module, keyword sets are acquired and weight-term pairs are produced as output. Section 3.1.1 discusses this module further.

- *Dimensionality Reduction* This module reads the dataset, finds a reduct and outputs the new reduced dataset. There are two implemented mechanisms to achieve this: rough set data reduction and entropy-based reduction. The operation of both have been highlighted in chapter 2.

- *Keyword Filtering* This uses the list of keywords generated by the dimensionality reduction stage to filter the keywords from the acquisition module.

- *Classification* The classifier uses the reduced dataset and the results from the previous module to classify the test data.

3.1.1 Keyword Acquisition

This module produces weight-term pairs given a dataset. All keywords are compared with the case ignored (e.g. 'house' equals 'HouSe'). Each of these keywords is assigned a weight according to its importance in the document, as determined by the following weighting methods:

- *Boolean Existential Metric.* All keywords that exist in the document are given a weight of 1, those that are absent are assigned 0.

- *Frequency Count Metric.* The frequency of the keywords in the document is used as the weight.

- *Term Frequency-Inverse Document Frequency Metric.* This assigns higher weights to those keywords that occur frequently in the current document but not in most others. It is calculated using the formula: $w(t,i) = F_i(t) \times log\frac{N}{N_t}$ where $F_i(t)$ is the frequency of term t in document i, N is the number of documents in the collection, and N_t is the total number of documents that contain t.

3.1.2 Rough Set Data Reduction

To be able to reduce a given dataset to its smallest possible dimensionality, a minimal reduct must be calculated somehow. The problem of finding such a reduct of minimal cardinality is, in general, NP-hard, and finding all reducts has exponential complexity. The following algorithm generates such a minimal reduct and is called the QuickReduct algorithm:

```
$R \leftarrow \emptyset$
$do$
  $T \leftarrow R$
  $\forall x \epsilon (\mathbf{C} - R)$
    $if \gamma_{R \cup \{x\}}(\mathbf{D}) > \gamma_T(\mathbf{D})$
      $T \leftarrow R \cup \{x\}$
  $R \leftarrow T$
$until\ \gamma_R(\mathbf{D}) = \gamma_{\mathbf{C}}(\mathbf{D})$
$return\ R$
```

(This algorithm is similar to the one presented in (Jelonek et al. 1995), but they use the calculated core as the initial R). At the start, R contains no attributes. The algorithm tries all the condition attributes not in R and adds to R the one that causes the largest increase in discernibility. The process is repeated until $\gamma_R(\mathbf{D}) = \gamma_C(\mathbf{D})$. It can be proven that the QuickReduct algorithm is monotonic, so it always locates a minimal reduct. The operation of the algorithm is best illustrated by returning to the example dataset in section 2.2. It calculates the dependency change after an attribute is added to R (initially empty):

$$\gamma_{R \cup \{a\}}(\{e\}) = 0/8$$
$$\gamma_{R \cup \{b\}}(\{e\}) = 1/8$$
$$\gamma_{R \cup \{c\}}(\{e\}) = 0/8$$
$$\gamma_{R \cup \{d\}}(\{e\}) = 2/8$$

Attribute d is added to R as it provides the highest increase in discernibility. The algorithm then repeats this previous step, but with $R = \{d\}$:

$$\gamma_{R\cup\{a\}}(\{e\}) = 3/8$$
$$\gamma_{R\cup\{b\}}(\{e\}) = 8/8$$
$$\gamma_{R\cup\{c\}}(\{e\}) = 8/8$$

By adding either b or c to R results in maximum discernibility. The algorithm will choose to add b to R as it always selects the left-most attribute when faced with two or more alternatives. So the reduct generated by the QuickReduct algorithm is $\{b,d\}$.

This method can be applied to any dataset. Once a minimal reduct is found, those attributes that are absent can be removed from the original dataset.

3.1.3 Entropy-Based Reduction

The rough set data reduction technique is reasonably well established and any alternative implementations of finding a minimal reduct are beyond the scope of this project. However the entropy-based approach has not been attempted before and so very little is known as to its applicability to this (or any other) domain.

Entropy-Based Reduction (EBR) is based on ID3 and was originally much closer to the ID3 approach than it is now. It followed the standard ID3 algorithm:

ID3(Examples, Target, Attributes)

1. Create a root node

2. If all Examples have the same Target value, give the root this label

3. Else if Attributes is empty, label the root according to the most common value

4. Else

   (a) Calculate the information gain for each attribute (using the entropy formula).

   (b) Select the attribute with the lowest entropy (call this A). Make A the tested attribute at the root

   (c) For each value v of A:
   Add a new branch below the root, corresponding to $A = v$
   Let Examples(v) be those examples with $A = v$

   If Examples(v) is empty, make the new branch a leaf node labelled with the most common value in the Examples

   Else let the new branch be the tree created by
   ID3(Examples(v), Target, Attributes - A)

The tree was constructed and then a breadth-first search was executed on the completed tree, returning an ordered list of the attributes. For large datasets, this required a lot of time (although not nearly as much as RSDR).

In the example in section 2.3, the final solution determined that a particular attribute was unnecessary. However, it is usually the case that all attributes appear in the final solution (but are ordered according to importance), so it is necessary to include an arbitrary threshold - some point at which the algorithm should halt and return those attributes that have been chosen so far. This can be adjusted to allow comparison with the RSDR approach.

It was found that by adopting a simpler approach to reduct generation, EBR could produce similar results to this old method. No tree construction occurs - it simply calculates the information gain for each attribute in the dataset. The attributes are then ordered accordingly, and the first *threshold* percent of them are considered to be the reduct. Returning to the example dataset in the previous chapter, the operation of ID3 was illustrated using table 2.3, and the following values were obtained:

$E(sickness) = 0.60684$
$E(day) = 0.5$
$E(time) = 0.75$
$E(weather) = 0.93872$

Using EBR, a different reduct is obtained. The attributes are ordered by the algorithm into: $\{day, sickness, time, weather\}$. With a threshold of 3/4 (we want three out of the four conditional attributes) the reduct $\{day, sickness, time, mood\}$ is produced. Using this method, a reduct of any length can be generated for any dataset.

3.1.4 Classification Modules

A number of different techniques were used to classify the test data:

Boolean Inexact Model

The problem with the *Boolean Exact Model* (described in section 2.1.1) is that it is inflexible - only documents for which the rule returns true match the rule. The *Boolean Inexact Model* (*BIM*) (Salton 1982) bypasses this problem by providing a scoring mechanism so that the rule with the highest score classifies the document. If there is more than one rule that fires then all rules must agree on the classification. If there is a conflict, then the classification is undecidable (and is classified into the 'Unclassified' category).

The main advantage of BIM is that it is fast (the computations involved are simple). It can also be quite accurate. A drawback is that words can have many meanings - something that the BIM cannot differentiate.

Vector Space Model

As mentioned in section 2.1.2, of all the TC techniques, the VSM is the most popular (Moukas and Maes 1998),(Salton et al 1975),(van Rijsbergen 1979). The vector space model procedure can be divided in to three stages. The first stage is the document indexing where content bearing terms are extracted from the document text. The second stage is the weighting of the indexed terms to enhance retrieval of documents relevant to the user. The last stage ranks the document with respect to the query according to a similarity measure.

Document Indexing : It is obvious that many of the words in a document do not describe the content, words like *the*, *is*, etc. By using automatic document indexing those non significant words (function words) are removed from the document vector, so the document will only be represented by content bearing words (Salton 1983). This indexing can be based on term frequency, where terms that have both high and low frequency within a document are considered to be function words

Term Weighting : Term weighting has been explained by controlling the exhaustivity and specificity of the search, where the exhaustivity is related to recall and specificity to precision. The term weighting for the vector space model has entirely been based on single term statistics. The three main factors in term weighting are *term frequency*, *collection frequency* and *length normalization*. These are multiplied together to make the resulting term weight.

Similarity Coefficients: The similarity in vector space models is determined by using associative coefficients based on the inner product of the document vector and query vector, where word overlap indicates similarity. The inner product is usually normalised. The most popular similarity measure (and the one used in this project) is the cosine coefficient, which measures the angle between the document vector and the query vector, and is defined as (taken from (Meadow 1992)):

$$Sim(X,Y) = \frac{|X \cap Y|}{\sqrt{|X|}\sqrt{|Y|}}$$

Fuzzy Reasoner

The previous chapter introduced the fuzzy rule-based technique for text categorisation. A document is classified using fuzzy rules with fuzzy reasoning (Zadeh 1965; Kasabov 1996; Cox 1994). All precondition memberships are evaluated, the fuzzy 'and' operator applied, and the rule with the highest score classifies the document (unless other rules match). Again, a set of firing rules is used and any inconsistencies are treated in the same way as the *BIM*.

3.2 Available Information

With the problems mentioned in the previous chapter in mind, we consider the classification of the data contained in the bookmark. An individual bookmark consists of many parts. In Netscape, a bookmark contains the following information:

- **URL** - the location of the document
- **ADD_DATE** - an integer that represents the number of seconds elapsed since midnight January 1, 1970
- **LAST_VISIT** - an integer that represents when the document was last visited.
- **LAST_MODIFIED** - an integer that represents when the document was modified last.
- **title** - the title of this document

In Internet Explorer, favorites are stored in directories, whose structure is the classification of those favorites. For example, favorites stored in "`/favorites/Sport and Leisure/Football/`" have the classification "`/Sport and Leisure/Football/`". The favorites themselves are saved as individual files (whose name is the title of the referred-to document) and contain the following information:

- **URL** - the location of the document
- **Modified** *(optional)*- when the document was last modified
- **DEFAULT***(optional)* - the base URL of this document.

Having looked at the information that is available for both Internet Explorer favorites and Netscape bookmarks, it can be seen that not all components of a bookmark/favorite are useful in the classification process. In fact only the *URL* and *title* contain helpful information, as the other components are time-related (except for *DEFAULT* which contains the same information as the *URL*). It is also beneficial to remove these attributes from the classification task to reduce the time taken for data reduction.

So the classification of bookmarks is carried out using only the information in the *URL* and *title* components. However, using a combination of these fields might produce an incorrect classification. For example "*ed*" in the *URL* part (referring to Edinburgh University) may well have a different meaning in the *title* part (probably a page belonging to a person named "*Ed*"). Classification, then, is performed on the *URL* and *title* fields separately, with the two results combined in some way to produce an overall category.

There are other problems when classifying a bookmark. First of all, the information in the *URL* or *title* fields can be minimal and not particularly informative. An example of this is "http://www.cnn.com" (with the imaginative title "CNN.com") which is a news site, but there is nothing to suggest this in the *URL* or *title* fields. There is also the frequent tendency for web sites to have the title “Home Page” for both commercial and personal sites.

There is also the problem of ambiguous information contained within the *title* fields. A page entitled “Java” could be talking about the programming language, the island, or a fern. Certainly the URL could be helpful in determining the context, but it is not guaranteed that any more information can be discovered as to what category the page belongs to.

3.3 Summary

In this chapter, the theoretical aspects integral to the system have been put forward. The system’s design is modular, allowing components to be changed easily for experimentation purposes and for future development. Two different methods for reducing datasets have been presented as have three different classification models. Such variety is necessary as one approach may be particularly suited to this domain; a key factor if the system is to perform well, given the small amount of information that is available.

Chapter 4

Software Specification

The details of the implementation of this system are presented here and defended. Many implementational decisions were enforced as a result of the language chosen and the application domain. This chapter seeks to relate the theory previously discussed with these decisions.

4.1 Choice of Language

It was decided from early on to implement the system in the Java programming language (Java 2000). Java was chosen as it is designed to operate on any platform (hence its slogan "write once, run anywhere"), and a bookmark utility such as this project can be used on any computer that has web access and a browser. Additionally, it is secure and enforces object-oriented design, leading to robust, well-structured and reusable code.

This project is based on earlier work carried out by Olve Maudal (Maudal 1996) and Alexios Chouchoulas (Chouchoulas 1999). The work was written originally in C++ due to its efficiency and modularity, and then additional code written in Perl. Part of the task of this project was to port some of the existing code from these languages to Java.

However, for all the advantages of the Java language it remains much slower than C++ at present (although in a few years time it is conceivable that Java could out-perform C++). The only time-consuming part of this system is the rough set reduction module which takes place during the training phase anyway (where time is not a problem). Almost all of the system is implemented in Java, with just one data-flow co-ordinating sub-system implemented in Perl (which is also portable).

4.2 System Implementation

4.2.1 Datasets

A large training database of bookmarks was required to ensure that the classifiers generated reasonable results. Fortunately, there are many sites on the internet where personal bookmarks are available to download (Web Wizards 2000; Dolphin 2000). The collection of individual bookmarks were standardised so that there were no duplicates and all equivalent categories had the same title. These were then combined to form a large bookmark hierarchy covering popular categories (such as "Arts", "News" etc). A maximum tree depth of 3 was chosen, as a deep hierarchy results in less efficient information retrieval.

4.2.2 Keyword Acquisition

Given a bookmark hierarchy (whether Internet Explorer favorites or Netscape bookmarks), this module acquires and weights the keywords present. As mentioned earlier, the *URL* and *title* are treated separately:

- *URL*: Keywords in a URL are those terms separated by white space characters or any of {-_+":/.()}. The case of each item is ignored.
- *title*: In a title, keywords are defined to be those terms separated by white space characters or any of {-+":;,.-[]{}()?/!@#$%$\hat{\&}$*= ''_}.

Common words such as 'the', 'is', 'and' etc are not considered. Again, the case of each item is ignored.

For each of these terms a weight is calculated and assigned to it according to the weighting method used. The three options are: *Boolean Existential*, *Frequency Count* and *Term Frequency-Inverse Document Frequency* metrics. These are defined in the previous chapter.

In the email categorisation system (Chouchoulas 1999), keyword weights were multiplied by a number depending on the field they were found in (i.e. words found in the "from" part of the message are considered more important than those found in the message body). It was decided that this was not appropriate for the bookmark categorisation system, as no real distiction of importance can be made between the terms. However, it can be seen that the title of a bookmark usually contains more classificatory information than the URL. This distinction is used in the classification phase (section 4.2.4).

The output produced by this module (in the training stage) is:

1. *Attribute Lists* - two lists (one for the URLs and one for the titles) of keywords and their corresponding weights as calculated by the weighting method. These lists also represent the column labels for the two tables of rules.

2. *Rule Tables* - two tables (one for the URLs and one for the titles) that can be used to classify data, but are not yet reduced. Each URL/title in the training database maps onto one row in the table. Each element at (row i,column j) indicates the weight of the keyword j in URL/title i. The very last column is an integer representing the classification.

3. *Classification List* - a list of the classifications with their corresponding integer representation. This list is neccessary, as the current implementation of the RSDR module only allows floating point numbers and not strings in the datasets.

4.2.3 Data Reduction

This module takes the rule tables resulting from the keyword acquisition module, finds a reduct (a list of column numbers considered to be the most useful for classification), reduces the rule tables accordingly, and sorts the keywords and corresponding table columns in order of weight (largest weighted keyword is the first column). Two data reduction approaches have been implemented:

Rough Set Data Reduction

The algorithm used to determine the minimal reduct is the QuickReduct algorithm (outlined in section 3.1.2). Maudal's (Maudal 1996) C++ implementation of this (called HAIG) has been converted to Java. However, as expected, the Java version is considerably slower than its C++ equivalent for large datasets. For this reason, the original program HAIG was used to speed up the process. In the future, the Java version may become preferrable, but at the moment it takes too long. Due to the modularity of the system, it is possible to include different implementations of the RSDR mechanism in different languages (so long as the computer supports the language), so it is quite simple to switch between the Java or C++ versions.

The HAIG program itself is not the most efficient implementation of QuickReduct. It requires a lot of memory (something that the Java implementation struggles with) and has a fairly high complexity. A highly optimised version of the RSDR mechanism has been implemented by Alexios (Chouchoulas 1999), but has not been converted to Java due to lack of time (it has not been fully tested yet either).

Entropy-Based Reduction

The operation of the Entropy-Based Reduction (EBR) technique has been outlined in chapter 3. It calculates and outputs a 'reduct' in a similar format to the RSDR approach. Given a dataset, EBR calculates the information gain for each attribute in the dataset and orders them, with the attribute that provides the most gain in information first. The first *threshold* percent of these attributes are returned as the data reduct.

In terms of speed, this approach is significantly faster at finding a reduction for datasets - it completes in a couple of minutes what it takes C++ HAIG several hours. Of course, this is a minor issue compared to the issue of classification precision. This is discussed in the next chapter.

Reducing the Datasets

As two rule tables have been generated, two reducts are generated by the reduction mechanism. These reducts are then used to reduce their corresponding rule tables. For both the URL and title tables, the following is the resultant output of this module:

- *Reduced Attribute List* - the same list generated by the keyword acquisition module, but with those keywords not present in the reduct removed. If the reduct {3,20,330,597,1200} was obtained, only the keywords representing columns 3, 20, 330 and 597 would be present (the final value is the decision column). Additionally, this list is sorted in order of decreasing weight.

- *Reduced Rule Table* - those columns of the original rule table not specified in the reduct are removed (in the example above the table would have only five columns). The columns are ordered according to the sorted reduced attribute list.

4.2.4 Classification

This module attempts to classify a given bookmark or bookmarks using the reduced rule tables and attribute lists. Each bookmark is examined and the URL and title determined. Each of these is then transformed using the keyword acquisition process defined earlier, but only those keywords present in the reduced attribute list are considered. Once converted, they are then classified individually and the results compared. Three different inference techniques were implemented:

- *Boolean Inexact Model* - this uses Boolean matching and scoring techniques. If a term exists in the URL/title and also exists in the corresponding rule table, then the score is increased. Also, if a term is absent from the URL/title and is also absent in the rule table, the score is increased. The rule with the highest score classifies the URL/title.
- *Vector Space Model* - this computes the similarity of a URL/title by calculating the angle between URL/title vectors.
- *Fuzzy Reasoner* - this follows the usual approach for the construction of fuzzy rule-based systems (Kasabov 1996). Reasoning is carried out by the fuzzy classifier using the rule table generated previously.

The operation of these models is defined in section 3.1.4.

To improve accuracy, it was decided that two classifiers should be used and the results merged. This leads to a total of four classifications for one bookmark as the classifiers operate on both the URL and title. For example, given a bookmark:

URL : http : //news.bbc.co.uk/ title : BBCNews|FrontPage|frontpage

One classifier might classify the URL as */News* and the title as */News/Newspapers*, while the other classifier might have the URL as */Recreational/TV* and the title classified as */Computers/*. The algorithm for deriving a final classification is very simple:

1. If at least two classifications match, then that is the classification of the bookmark.
2. Else find the lowest classification directory where at least two classifications are the same. In the example this would be */News* as */News* and */News/Newspapers* have this as a common directory.
3. Else the bookmark is unclassified.

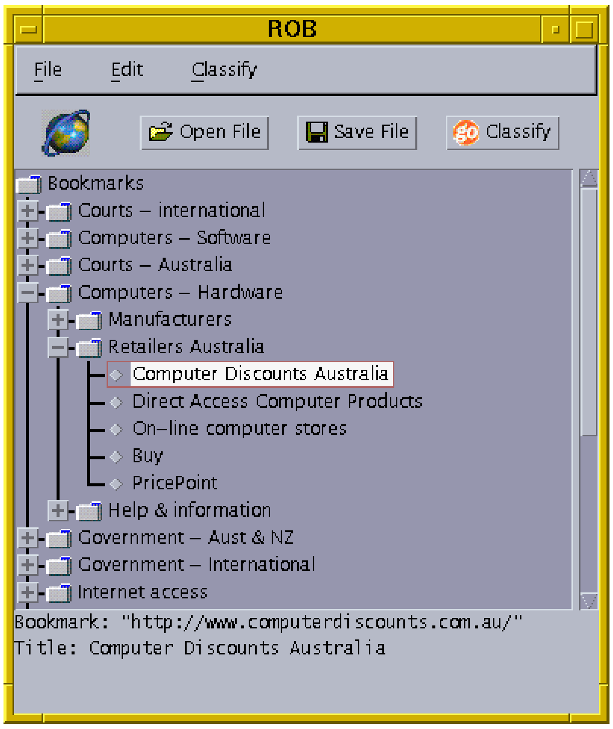
4.3 User Interface

To show that the system is viable as a utility to manage and automatically classify a users' bookmark collection, a simple user interface was constructed. This system is experimental - there are many features missing that should be present in a proper application. The idea here is to demonstrate that such an application can be constructed. The working title of this application is Rob - Roughly Ordered Bookmarks.

Bookmarks are displayed in an expandable tree structure similar to that used in Netscape itself. The program can read and display both Internet Explorer favorites or Netscape bookmarks. It can also save to either format, allowing the user to automatically organise their collection and keep the results. For simplicity, only those bookmarks at the top level in a user specified folder are automatically classified. An example of this system in action can be found in chapter 5.

Features that a complete application should have are:

- *Drag and Drop* - the user should be able to click and drag bookmarks into or out of folders, folders into or out of other folders, etc.



Bookmarks
- Courts – international
- Computers – Software
- Courts – Australia
- Computers – Hardware
  - Manufacturers
  - Retailers Australia
    - Computer Discounts Australia
    - Direct Access Computer Products
    - On–line computer stores
    - Buy
    - PricePoint
  - Help & information
- Government – Aust & NZ
- Government – International
- Internet access

```
Bookmark: "http://www.computerdiscounts.com.au/"
Title: Computer Discounts Australia
```

Figure 4.1: Examining Explorer Favorites with Rob

- *Classify Individual Bookmarks* - the system at present classifies all bookmarks at the top level of a folder, but not individual ones. This is quite straight-forward to code, but due to lack of time it was not implemented.

- *User-Defined Categories* - the user should be able to rename the exisitng categories into different ones more relevant to the user.

- *Learning* - the ideal system should be able to learn new classifications. This is discussed in the final chapter.

4.4 Summary

The choice of Java as the implementational language has provided many benefits (it is robust, secure and portable), but has also brought with it a disadvantage - it takes too long to calculate a reduct in comparison with C++. This is not seen as too problematic, as machines are getting faster, and the Java programming language is being developed further and becoming more efficient.

Various implementational details have been discussed, relating to the individual processes involved. The Keyword Acquisition phase obtains sets of weight-term pairs for each bookmark, converting this to two table from which bookmarks can be classified. The Data Reduction module reduces these tables whilst attempting to retain classification precision. Finally, the classifiers attempt to produce accurate classifications of new bookmarks using these reduced datasets.

Chapter 5

Experimentation

The original aim was to investigate how well RSDR can extract information from the bookmark domain. The experiments presented here attempt to test whether RSDR is a useful tool for reducing data whilst retaining the information content. Additionally, experiments are carried out that compare the performance of RSDR with that of an alternative approach, EBR. The classification models are combined in order to classify the bookmarks, each combination is investigated here also.

5.1 Experiments

5.1.1 Performance Measure

As mentioned in (Chouchoulas 1999), Information Filtering and Information Retrieval systems are compared using the concepts of *precision* (the number of relevant retrieved documents out of the total number retrieved) and *recall* (the number of relevant documents retrieved out of the total number of relevant documents known to the system). However in this system, documents are not considered to be relevant or irrelevant, but may be classified into a number of classes. For this reason, performance is measured by how well the system classifies bookmarks (i.e. if the categories suggested by the system are appropriate).

5.1.2 A Problem

Before the results are given, a problem encountered during this phase needs to be highlighted. The collation of results has to be undertaken manually as bookmarks may have many valid but differing classifications. For instance a bookmark about sports news can be classified either as belonging to the sports category or the news category. A large number of bookmarks exhibit this problem and there is no automatic way of differentiating between an incorrect classification and an unexpected, but correct one. So, all classifications have to be manually verified.

A much greater amount of experimentation is required, but due to its time-consuming nature, has not been carried out yet.

5.1.3 Testing the Goal

The principle goal of this project was to demonstrate the effectiveness of using rough sets to reduce dataset dimensionality, while retaining the information content. To test this, a comparison of classification precision between rough set reduced datasets and unreduced datasets was set up. The unreduced method simply used the generated rule tables from the stage before the data reduction module. There has been some reduction though, as stop words have been removed. The RS-reduced method used the reduced datasets from the RSDR process.

For the test data, a subset of the original training bookmark examples were randomly selected, covering all categories. If the RS-reduced dataset contains the same information content as the unreduced dataset, then the classification of this set of bookmarks should be identical (or close to it). As mentioned before, the use of just one classifier method proved too inaccurate, so pairs of classifiers were used to classify the test data. The results are displayed in table 5.1. (VSM is the Vector Space Model, BIM is the Boolean Inexact Model, and FR is the Fuzzy Reasoner).

Dataset	Attributes (URL)	Attributes (Title)	VSM + BIM	VSM + FR	FR + BIM
Unreduced	1397	1283	98.1%	96.8%	98.7%
RS-reduced	514	424	94.3%	94.9%	98.1%

Table 5.1: Comparison of Unreduced and RS-reduced classification accuracy

From the table we can see that using rough set theory, the amount of attributes was reduced to around 35%. For email classification, the average reduction of attributes was 3.5 orders of magnitude. This demonstrates that there is much less redundancy in the original datasets for the bookmark domain, which is intuitive as there is much less information in a bookmark than a document.

The greatest drop in classification precision was 4% for the combination of the Vector Space Model and the Boolean Inexact Model. Even this loss of precision is acceptable when considering the increase in speed gained from a reduced dataset. The combination of the Fuzzy Reasoner and the Boolean Inexact Model give the best results here, perhaps because of their quite different approaches. The Vector Space Model was observed to produce similar classifications as the FR before these experiments were carried out, so it was expected to perform similarly. This appears to be the case.

It was expected that the unreduced datasets would produce perfect results (i.e. 100% precision), but this was not the case. An explanation for this might be that other rules produce the same score as the correct rule, but the conflict resolution strategy chooses the wrong rule to classify the bookmark. There is also a degree of variance between the classifiers themselves - some appear to better than others for this task. Having said this, most of those classifications deemed "incorrect" are acceptable alternatives from a users' point of view.

The same dataset was used for two other approaches: entropy-based reduction and random-reduct reduction (RR). RR generates a random reduct given the total number of attributes present in the unreduced table. The size of the reduct is constrained to be equal to the size of the rough set reduct. Using this reduct, the dataset is reduced and used for experimentation. The results of these two approaches can be found in table 5.2.

Dataset	Attributes (URL)	Attributes (Title)	VSM + BIM	VSM + FR	FR + BIM
EBR-reduced	514	424	72.8%	72.8%	74.7%
RR-reduced	514	424	49.4%	50.0%	52.5%

Table 5.2: Comparison of EBR-reduced and RR-reduced classification accuracy

The RR-reduced dataset performs as expected, with an accuracy of around 50%. The EBR-reduced dataset didn't perform as well as the rough set alternative, with a comparative drop in accuracy of around 20%. EBR and RSDR are compared more thoroughly in the next section.

5.1.4 EBR vs Rough Sets

Several datasets have been obtained from the internet. Each bookmark dataset is someone's personal collection that has not been used in the training phase and have not been modified in any way except to remove any duplicate entries. They have been chosen to simulate how this system would cope with a user's real bookmark database.

The collections contain a flat list of bookmarks, the contents of which are filtered using the reduct generated by the data reduction module. The resulting keyword set is then used by the classifiers to generate a classification. The overall results are displayed in table 5.3.

Dataset	VSM + BIM	VSM + FR	FR + BIM
Unreduced	55.6%	49.7%	45.0%
RS-reduced	49.1%	47.3%	42.0%
EBR-reduced	50.9%	52.7%	43.2%
RR-reduced	37.3%	34.9%	26.3%

Table 5.3: Comparison of reduction strategies with unreduced dataset

We have seen that in table 5.1, the combination of the FR and BIM classifiers produced the most accurate classification of the training data. In table 5.3, however, we see that they are the least accurate classifiers for new data and in fact VSM/BIM was the most successful combination. The BIM immediately notices the existence of any known keywords and classifies them accordingly; for instance, if the title contained the word "java", the BIM would recognise this as belonging to the "Java" category. As titles tend to have only the important keywords present, the BIM was expected to perform well. The results obtained are inconclusive on this issue.

The table shows that the overall accuracy is poor (obviously, the random reduction gives weak results). The main point to make here is that the ability of the datasets to classify new data depends entirely on the quality (and to a certain extent the quantity) of the training data. It can't be expected that the RS-reduced or the EBR-reduced experiments should perform better than the unreduced dataset.

Performance of all the methods employed could be improved by reducing the number of sub-categories (thus reducing the chance of mis-classifying a bookmark). For example, categories such as $/Computers/Reference/Perl$ could become just $/Computer$. For large bookmark databases this is unacceptable as each category will contain a huge list of bookmarks, defeating the whole point of having an automatic bookmark classification system.

In light of the fact that bookmarks contain very little useful information, the results are unsurprising and perhaps a little better than anticipated. As stated earlier, the goal of this project is to investigate how useful rough set theory is in reducing the training dataset. For this, we need to compare how well the rough set-reduced approach fares against the unreduced dataset. If we consider the unreduced dataset results to be the optimum, the table can be rewritten (shown in table 5.4).

Dataset	VSM + BIM	VSM + FR	FR + BIM
RS-reduced	88.3%	95.2%	93.3%
EBR-reduced	91.5%	106%	96.0%
RR-reduced	67.1%	70.2%	58.4%

Table 5.4: Comparison of reduction strategies

Viewed this way, we can see that EBR has the best results for each classifier pair, and is in fact better than the unreduced dataset in one instance. The performance of the RS-reduced dataset is almost as good. From this we can conclude that a very small amount of important information has been lost in the rough set reduction approach, but not enough to reduce classification accuracy significantly. The results appear to show that EBR works best for unseen data, but RSDR is best for known examples.

The question now asked is: why should EBR perform better than the "optimum" for new data? EBR selects those attributes that provide the largest gain in information. This process might ignore otherwise misleading attributes that the unreduced dataset contains. The RS-reduced dataset can be thought of as a smaller version of the original dataset, and so this will fall prey to the same mistakes.

A more thorough investigation into the performance of EBR compared to

RSDR would have been desirable, to ensure that these results were accurate, but there was not enough time for this. It might possibly be that EBR is particularly suited to the bookmark domain for some reason that is not obvious. Appendix A contains some results from using EBR on datasets from other domains, and results are presented alongside those from RSDR.

5.2 Sample Run

5.2.1 Training

This section details the actions required to convert the input data (in the form of bookmarks) into a reduced training dataset that can be used by the classifiers.

A Perl script has been written that takes a database of bookmarks, converts them to a format usable by the data reduction phase, finds a reduct and then reduces the dataset accordingly. To produce the rule tables for the bookmark collection in the file "bookmarks.html", the following command is used:

```
perl run.pl bookmarks.html training -f -n
```

which stores the necessary data in the specified directory "training", using the frequency count metric. The final argument ("-n") indicates that this is a Netscape bookmark collection; with the option "-e", a directory of favorites needs to be specified instead of the filename "bookmarks.html", for example:

```
perl run.pl /windows/favorites training -f -e
```

The script then enters the following phases in order:

Bookmark Conversion

The bookmark file is examined and the following information is obtained and saved to separate files:

- *Classifications*: The classification of each URL is stored. In Netscape the classification is equivalent to where the URL is in the HTML directory structure, whereas in Explorer the classification corresponds to the path from the root directory to the URL.

- *List of Title Keywords*: A list of all the text contained in the title fields. Each line in this file contains the string of words determined by the module to be useful. The set of words on line 1 corresponds to the text in the first bookmark, etc.

- *List of URL Keywords*: Similar to the list above, this contains all the keywords found in the URLs.

Acquisition of Keywords

Both the keyword files generated by the previous phase are processed here. An additional parameter is required to specify which of the weighting methods to use (*Boolean Existential*, *Frequency Count*, or *TF-IDF*). The program then reads each file individually and generates the following:

- *Keyword-Value Pairs*: A list of all the keywords present and the number of times it occurs in the database. Each line corresponds to a separate keyword. This is saved as the file `p_attr.txt` or `p_attr.url`.

- *Partial Dataset*: This is a table whose columns map to the keywords in the keyword-value pair list (first column = first keyword), and whose rows are generated by each bookmark (first row = first bookmark in the database). The value at (row i,column j) is the weight for keyword i in bookmark j calculated using the specified weighting method. This table is incomplete as it does not yet contain the corresponding classifications. This is saved as the file `p_table.txt` or `p_table.url`.

Finalising

The final phase examines the classification file and constructs a new file (called `table.classes`), which contains each distinct classification and an associated identifying number. Using this, the classifications (in numerical form) are added to the partial tables, resulting in the complete tables (called `t.txt` and `t.url`). These tend to be very large in size (around 3.5 Megabytes for a decent training database). It is these tables that can now be reduced by the data reduction mechanism.

Data Reduction

The datasets generated previously are reduced in this phase. There are two implemented data reduction mechanisms: RSDR and EBR.

- *RSDR*: To find a reduct using rough set theory, the program HAIG needs to be executed (on both the URL and title tables) as follows:
  ```
  java haig <dataset>
  ```
  where $< dataset >$ in this example will be `training/t.txt` for the title table and `training/t.url` for the URLs.

- *EBR*: To discover a reduct by means of the entropy-based reduction technique, the program *ebr* should be executed on both tables:
  ```
  java ebr <dataset> <threshold>
  ```
  In this example, the specified dataset is `training/t.txt` or `training/t.url`. The floating-point number $< threshold >$ can be specified (if not, the default value is 0.4 corresponding to 40%). This value represents how many attributes (as a percentage of the total number of attributes in the dataset) should appear in the reduct. In order to compare *EBR* with *RSDR*, the proportion of attributes in the *RSDR* reduct was calculated and used as the threshold value; this enforced *EBR* to produce a reduct of identical cardinality. For large datasets (more than

about 50 attributes) it was observed that the reduct size for *RSDR* was between 30% and 40% of the original number of attributes, hence the default threshold value of 40%.

The result of both of these processes is a list of attributes that are considered to be vital for the classification task at hand. An example reduct calculated from the iris dataset (ML 2000) is:

`java haig iris.dat` = {0,1,2,4}
`java ebr iris.dat 0.8` = {0,2,3,4} (also a viable reduct)

Using *RSDR*, column 3 would not appear in the final table. If *EBR* is used, column 1 would be removed instead.

Those attributes not appearing in this list are then removed from the tables by a separate program named *reducer*, which produces the following files:

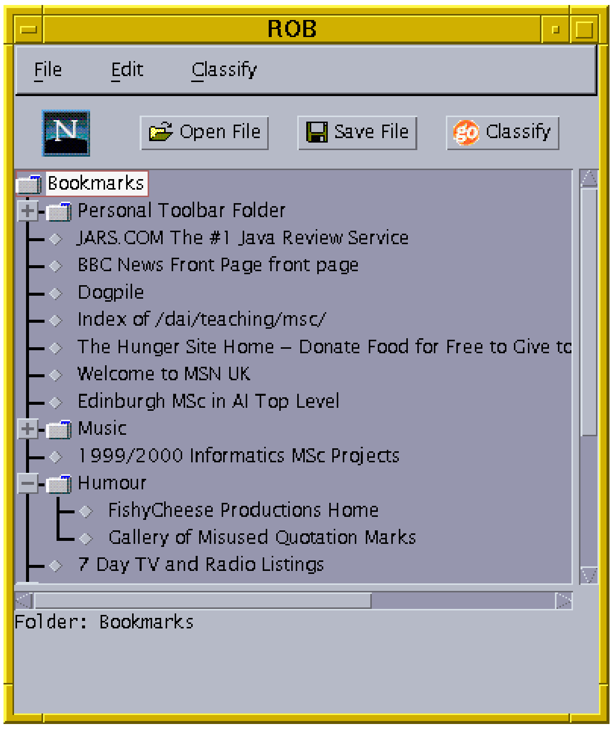
`table_txt.rea`, `table_url.rea` - the reduced list of attributes for the URL and title tables.
`table_txt.red`, `table_url.red` - the actual reduced datasets.

These files can now be used by the classification module to classify new bookmarks.

5.2.2 Testing

This section seeks to illustrate the process of classification by going through a typical example using the user interface and then detailing the modules involved.

The user starts up Rob and selects their Netscape bookmark file (any file can be chosen if it contains bookmarks), and the expandable tree structure



Bookmarks
Personal Toolbar Folder
JARS.COM The #1 Java Review Service
BBC News Front Page front page
Dogpile
Index of /dai/teaching/msc/
The Hunger Site Home – Donate Food for Free to Give tc
Welcome to MSN UK
Edinburgh MSc in AI Top Level
Music
1999/2000 Informatics MSc Projects
Humour
FishyCheese Productions Home
Gallery of Misused Quotation Marks
7 Day TV and Radio Listings
Folder: Bookmarks

Figure 5.1: An example bookmark file

representing the hierarchy is displayed. As mentioned previously, the user specifies a folder and clicks on *Classify* to automatically re-categorise the top-level bookmarks in that folder. In this example, they have selected the top folder (the root of the tree). The user clicks on *Classify* and then waits for the program to automatically categorise the top-level bookmarks. The tree has now changed:

From further examination, the user can locate where each bookmark has been reclassified:

Classification

For a bookmark to be classified, it must first be converted into a format that can be manipulated and used by the classifiers. Using the selected folder from

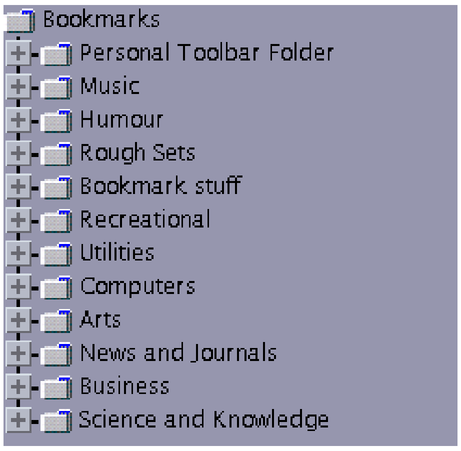


Figure 5.2: Sorted bookmarks

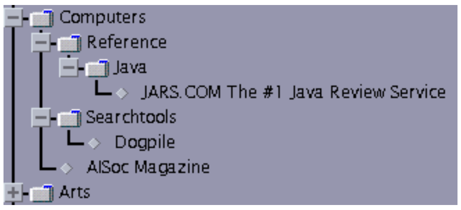


Figure 5.3: Exploration of classifications

the user interface, each top-level bookmark is retrieved and the URL and title fields are examined. Keywords are acquired from these fields and weighted according to the weighting method (if they exist in the reduct generated by the data reduction module). The document is now represented as an array of floating-point numbers where `document[ind]` returns the weight of the keyword with index *ind*. This is now in a format that can be used to classify the bookmark.

The bookmark arrays (for the URL and title independently) are compared to every rule in the rule table. A score is calculated according to the classification model used, and the rule with the highest score returns its associated classification number. This value is used to locate the string version of the classification.

As mentioned in chapter 4, two classifiers are used to make the final decision by comparing the results of their URL and title classifications. Although this slows the process down, it makes for greater classification accuracy, which is essential when attempting to classify something as information-poor as a bookmark.

After all classifications are obtained, the tree is updated accordingly. The user is free to traverse the hierarchy and classify other bookmarks.

5.3 Comparison with Existing Systems

This system aims to incorporate most of the advantages of existing automatic organisers of bookmark, whilst avoiding the disadvantages they possess.

5.3.1 Bookmark Organiser

The first system looked at was the Bookmark Organiser, which classified bookmarks by the text in the documents that they referred to. This is obviously going to be more accurate, as the document itself contains considerably more information than its associated bookmark.

Bookmark Organiser generated a reasonably shallow folder hierarchy. This is important for quickly locating bookmarks and not making mistakes finding them. This feature is present in Rob, where the depth of the hierarchy is limited to a maximum of three folders (e.g. "/Computers/Reference/Java").

The other useful feature in BO is the fact that it allows fully-automatic and semi-automatic classification. A user can use the tool to manually put bookmarks into the appropriate categories, or can request BO to do this for them. In Rob, fully-automatic classification has been implemented but there was not enough time to implement a drag-and-drop semi-automatic facility. This is just a side issue, however, as the project is experimental and concerned with the usefulness of rough set theory, not the design of the interface.

The main limitation of the Bookmark Organiser was that the automatically generated categories had unclear names. This is quite a fundamental flaw for a system that is designed to facilitate the maintainence of a meaningful bookmark hierarchy. A user with no formal background in computer science will not be able to make sense of the folder titles. In Rob, all classifications have meaningful titles, with the top-level categories covering the main topics (such as “Art”, “Business”, “Health”, etc). These were based on such internet directories as AltaVista and Yahoo (Altavista 2000; Yahoo 2000).

It would have been interesting to compare the speed and accuracy of BO and Rob for identical testing datasets, but there has been no analysis to this effect in BO, nor is there a downloadable version available with which experimentation can be carried out.

5.3.2 PowerBookmarks

It must first be pointed out that PowerBookmarks was intended for a different environment, where multiple users have their own bookmarks and additionally can share them. This results in different requirements and design decisions than those of a single-user application.

The classification process used by PowerBookmarks is similar in some ways to BO as the referred-to document is examined instead of the bookmark itself. Metadata is derived from the document based on keyword frequency analysis. This metadata is used by external classifiers (such as an Internet search engine) to generate the classification. Like Rob, the categories produced are limited to a certain depth and have manually-selected labels.

The problem with PowerBookmarks is that all a user’s bookmark operations must be carried out online. This is not practical for the ordinary home-based user. ROB was designed as a utility for such a user, and can be run on any machine (due to its Java implementation) to automatically sort bookmarks.

Chapter 6

Conclusion

The initial aim of this project was to investigate how well rough set theory can help extract information from a relatively information-poor domain; the databases of 'bookmarks' or 'favorites' saved by WWW browsers. This chapter evaluates the success of this system in achieving this aim as well as discussing the implemented extensions. Finally, it concludes with some suggestions for future work.

6.1 Project Evaluation

Results clearly show that rough set theory can be used to significantly reduce the dimensionality of the training dataset without much loss in information content. The measured drop in classification accuracy was between 0.6% and 4% for the training dataset, which is within acceptable bounds.

The system is modular, allowing different implementations of various components to be inserted and tested. An alternative data reduction mechanism, EBR, was constructed and compared with the RSDR approach. EBR looks promising in this area (it betters RSDR in classifying unseen data), but further experimentation is required to ensure the validity of these results. Alternative classification models were implemented - the best one for classifying unseen data was the vector space model combined with the boolean inexact model.

6.1.1 Limitations

The system compares well with other existing systems, improving on their limitations and building on their advantages. As an overall application, the system is far from ideal though. A classification accuracy of around 50% means that half of all new bookmark classifications are incorrect - which is not good enough for a commercially-viable system. This system has always been considered an experimental one, mainly due to the near impossible nature of classifying a bookmark from such little information as its URL and title. In this respect, a classification accuracy of 50% is about as successful as such a system is ever going to be.

The main limitation of this system is that it will only be as good as the training dataset itself. Ideally, a much larger database of bookmarks would have been used, but this would have taken many more weeks of collation. It is not known how long it would take the QuickReduct algorithm to find a reduct for such a large dataset as it takes many hours to find one for the existing training dataset.

Another limitation is that only pre-defined categories can be generated by the system. If a user becomes interested in a new topic, then there is no way of creating a suitable category for these bookmarks and then automatically classifying similar bookmarks to this new category. Linked in with this is the fact that the system cannot learn, given new classifications. An option would be to keep the training database and add new classification rules to it whenever a user informs the system of this new classification. Every day or so, the training process could run, reducing the dataset for the classifiers to utilise.

6.1.2 Achievements

The main goal has been achieved - it has been shown that rough set theory is a useful tool for extracting information out of the bookmark domain. Classification accuracy is retained after the datasets have been reduced.

The success of EBR in generating useful reducts is a little surprising, due to its straightforward approach. As an alternative data reduction technique, it fares well against RSDR as shown in appendix A. However, with EBR a threshold needs to be specified beforehand. With no RSDR reducts to estimate this value, there is no method available for discovering the appropriate number of attributes that should appear. From experimentation with larger datasets, it appears that a value of around 35% of the total number of attributes is a reasonable estimate though. Another drawback with EBR is that it can never find more than one possible reduct, which is perfectly fine for applications such as this, but may not be for more theoretical investigations.

6.2 Future Work

As mentioned earlier in this chapter, the main problem posed is one of achieving acceptable accuracy. It has been shown through experimentation that there is not enough information in a bookmark to classify it with any degree of certainty. A better strategy would be to locate the document that the bookmark represents and use TC techniques on the text contained in that document (which is quite straightforward to do using Java). This would provide a more accurate classification, which could be compared with the classification of the URL and title to make a final decision. This would require access to the internet every time automatic classification is desired (unless the bookmarked web pages were stored locally).

Learning is a desirable feature that could be achieved in a laborious manner by adding new classifications to the training dataset. A new bookmark would be converted into the table format and inserted at the end of the table. The dataset could then be reduced using EBR for use by the classifiers (that should now be able to classify similar bookmarks correctly). There may be many other feasible ways of approaching this topic, such as using neural networks trained on the pre-classified data, reduced by RSDR. Another possibility could be an RSDR algorithm that incorporates training into its operation.

At the implementational level, further work could be carried out on Haig. At the moment it requires too much memory and is quite slow for large datasets. A highly streamlined version of Haig would be desirable, reducing the amount of time required in producing the training datasets. Many improvements need to be made to the user interface - its purpose was only to demonstrate that such a system is possible. It lacks some of the features necessary for a finished application, and these have been mentioned in chapter 4.

EBR has proven to be very successful in this domain (and others - see Appendix A). Therefore, further investigation into it's applicability to new domains should be carried out. Existing systems that employ a data reduction step could be tested, with EBR providing this functionality. Providing the system is reasonably modular, it should require very little effort to integrate EBR into the overall architecture. It is important to research this further as it is much faster than RSDR (for all the datasets investigated here) and appears to produce competitive results. The theory behind EBR should be developed also, as there may well be many improvements and additions that could be made to produce even better results.

Bibliography

[Altavista 2000] Altavista. A WWW-based Information Retrieval System or Search Engine: http://www.altavista.com/

[Apté et al 1994] Chidnand Apté, Fred Damerau, and Sholom M. Weiss. Automated learning of decision rules for text categorization. ACM Transactions on Information Systems, 12(3):233-251, 1994.

[Atardi et al. 1999] Giuseppe Attardi and Antonio Gullí and Fabrizio Sebastiani. Automatic Web Page Categorization by Link and Context Analysis. Proceedings of THAI-99, 1st European Symposium on Telematics, Hypermedia and Artificial Intelligence, 1999.

[Baltzersen 1995] Jorn Kristian Baltzersen. Quinlan's Algorithms and the Rough Sets Approach. Project Report, Department of Computer Systems and Telematics, University of Trondheim, 1995.

[Callan 2000] Jamie Callan. Vector Space Retrieval Model: Introduction. http://ciir.cs.umass.edu/cmpsci646/

[Chakrabarti et al 1998] Chakrabarti, S., Dom, B., Raghaven, P., Rajagopalan, S., Gibson, D., and Kleinberg, J. Automatic resource compilation by analyzing hyperlink structure and associated text. Computer Networks and ISDN Systems 30, 1-7, 65-74.

[Chouchoulas 1999] Alexios Chouchoulas and Qiang Shen. A Rough Set Approach to Text Classification, Proceedings of the Seventh International Workshop on Rough Sets (Lecture Notes in Artificial Intelligence, No. 1711), pp. 118-127, 1999.

[Cox 1994] Earl Cox. The Fuzzy Systems Handbook: a Practitioner's Guide to Building, Using and Maintaining Fuzzy Systems. Academic Press, Inc., 1994. ISBN 0-121-94270-8.

[Deerwester et al 1990] S. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, R.A. Harshman. Indexing by latent semantic analysis. Journal of the Society for Information Science, 1990.

[Dolphin 2000] Dolphin Systems Ltd. - "My Personal Bookmarks" http://www.dolphinsystems.co.uk/bookmark.htm

[Dumais 1996] Susan T. Dumais. Combining evidence for effective information filtering. AAAI Spring Symposium on Machine Learning in Information Access Technical Papers, 1996.

[EBRSC 1993] A Brief Introduction to Rough Sets, Copyright 1993, EBRSC. Information available at: http://cs.uregina.ca/r̃oughset/rs.intro.txt

[Fuhr 1992] Norbert Fuhr. Probabilistic Models in Information Retrieval. Computer Journal. 35 (3) , p. 243-55, 1992.

[Fuhr et al 1998] Fuhr, N., Gövert, N., Lalmas, M., and Sebastiani, F. Categorisation tool: Final prototype. Deliverable 4.3, Project LE4-8303 "EUROSEARCH", Commission of the European Communities, 1999.

[GVU 1997] Georgia Tech Research Corporation, GVU's 8th WWW User Survey, 1997, information available at: http://www.gvu.gatech.edu/user_surveys/survey-1997-10/

[Java 2000] The Java home page: http://java.sun.com.

[Jelonek et al. 1995] Jacek Jelonek, Krzysztof Krawiek, and Roman Slowinski. Rough set reduction of attributes and their domains for neural networks. Computational Intelligence, 11(2): 339-347, 1995.

[Kasabov 1996] Nikola K. Kasabov. Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering. The MIT Press, 1996. ISBN 0-262-11212-4.

[Larson and Czerwinski 1998] K. Larson and M. Czerwinski, Web page design: implications of memory, structure and scent for information retrieval, in: Proc. 1998 ACM SIGCHI Conf. on Human Factors in Computing Systems, Los Angeles, CA, April 1998, pp. 25-32.

[Li et al 1999] Wen-Syan Li, Quoc Vu, Divakant Agrawal, Yoshinori Hara, Hajime Takano. PowerBookmarks: a system for personalizable Web information organization, sharing, and management. Proceedings of the Eighth International World Wide Web Conference, Toronto, Canada, 11-14 May 1999, ISBN 0-444-50264-5.

[Maarek and Shaul 1996] Yoelle S. Maarek, Israel Z. Ben Shaul. Automatically Organizing Bookmarks per Contents. Fifth International World Wide Web Conference 1996, Paris, France. http://www5conf.inria.fr/fich_html/papers/P37/Overview.html

[Maudal 1996] Olve Maudal. Preprocessing data for Neural Network based Classifiers: Rough Sets vs Principal Component Analysis. Project Report, Department of Artificial Intelligence, The University of Edinburgh, 1996.

[Meadow 1992] Meadow, Charles T. Text Information Retrieval Systems. Academic Press, 1992.

[ML 2000] Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science:
http://www.ics.uci.edu/m̃learn/MLRepository.html.

[Moukas and Maes 1998] Alexandros Moukas and Pattie Maes. Amalthaea: An evolving Multi-Agent Information Filtering and Discovery System for the WWW. Journal of Autonomous Agents and Multi-Agent Systems, 1(1):59-88, 1998.

[Moulinier 1996] Isabelle Moulinier. A Framework for Comparing Text Categorization Approaches, Universite Paris, 1996. Proceedings of the 1996 AAAI Spring Symposium on Machine Learning in Information Access, 1996.

[Munakata 1998] Munakata, T. (1998). Fundamentals of the new artificial intelligence: Beyond traditional paradigms. Springer-Verlag.

[Ng et al. 1997] Ng, H. T., Goh, W. B., and Low, K. L. Feature selection, perceptron learning, and a usability case study for text categorisation. In Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval (Philadelphia, US, 1997), pp 67-73.

[Oard 1999] Douglas W Oard. The Vector Space Model, 1999.
http://www.clis.umd.edu/academics/courses/708a/summer99/notes/

[Pawlak 1982] Zdzislaw Pawlak. Rough Sets: International Journal of Computer and Information Sciences, 11(5); 341-356, 1982.

[Pawlak 1991] Zdzislaw Pawlak. Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishing, Dordrecht, 1991.

[Quinlan 1986] J.R. Quinlan. Induction of Decision Trees. Machine Learning 1(1), pp. 81-106. 1986.

[Ruiz and Srinivasan 1999] Ruiz, M. E. and Srinivasan, P. Hierarchical neural networks for text categorization. In Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval (Berkeley, US, 1999), pp 281-282.

[Salton et al 1975] G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. Communications of the ACM, 18(11):613-620, 1975.

[Salton 1982] Salton, Gerard, Fox, Edward A. and Wu, Harry, (Cornell Technical Report TR82-511) Extended Boolean Information Retrieval. Cornell University. August 1982.

[Salton 1983] Salton, Gerard. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.

[Sebastiani 1999] Fabrizio Sebastiani. Machine learning in Automated Text Categorisation: a survey. Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, 1999.
http://faure.iei.pi.cnr.it/f̃abrizio/Publications/ACMCS00/ACMCS00.pdf

[Shütze et al. 1995] Shütze, H., Hull, D. A., and Pederson, J. O. A comparison of classifiers and document representations for the routing problem. In Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval (Seattle, US, 1995), pp 229-237.

[Slowinski 1992] Roman Slowinski, editor. Intelligent Decision Support. Kluwer Academic Publishers, Dordrecht, 1992.

[Tauscher and Greenberg 1997] L. Tauscher and S. Greenberg, Revisitation patterns in World Wide Web navigation, in: Proc. 1997 ACM CHI Conference, Atlanta, GA, March 1997.

[van Rijsbergen 1979] C.J. van Rijsbergen. Information Retrieval. Butterworths, London, United Kingdom, 1979.
http://www.dcs.gla.ac.uk/Keith/Preface.html

[Web Wizards 2000] Useful Bookmarks - Content-rich web sites for everyday use: http://www.webwizards.net/useful/

[Yahoo 2000] Yahoo Web Directory and Search Engine.
http://www.yahoo.com/

[Zadeh 1965] Lofti A. Zadeh. Fuzzy sets. Information and Control, 8:338-353, 1965.

Appendix A

Further EBR investigations

In this appendix we look at how EBR reduces datasets from domains other than the bookmark domain. Reducts for the datasets are generated with EBR and then compared with those produced by RSDR. As RSDR takes a large amount of time to discover all reducts, only small datasets are being investigated here.

All datasets mentioned are available from the UCI ML Repository (ML 2000), except the Fruit and Mood datasets. To avoid over-complicating the results, individual details of each dataset are omitted - descriptions can be found at the ML Repository. Some dataset values were strings, so these were altered to a corresponding numeric value.

Reducts labelled “a reduct” indicate that this is one of many possible RS minimal reducts. Those labelled “EBR ¡number¿” have been generated by the EBR method, with threshold set to ¡number¿. Those labelled “only” reduct indicate that this is the only minimal reduct for the dataset.

Iris Dataset

a reduct = {0,2,3,4}

EBR 0.8 = {0,2,3,4}

Solar Flare Dataset

only reduct = {0,1,2,4,5,6,10,11,12}

EBR 0.7 = {0,1,2,4,6,9,10,11,12}

EBR 0.75 = {0,1,2,4,5,6,9,10,11,12}

Soybean Dataset (small)

a reduct = {20,21,35}

EBR = {20,21,35}

Shuttle Landing Control Database

reducts = {0,1,6}
$\{0, 2, 6\}$
$\{0, 5, 6\}$
$\{1, 5, 6\}$

EBR 0.4 = {1,2,6}

EBR 0.5 = {1,2,5,6}

Congressional Voting Records Database

reducts (all) = {0,1,2,3,4,8,10,12,15,16}
$\{0, 1, 2, 3, 5, 8, 10, 12, 15, 16\}$
$\{0, 1, 2, 3, 6, 8, 10, 12, 15, 16\}$
$\{0, 1, 2, 3, 7, 8, 10, 12, 15, 16\}$
$\{0, 1, 2, 3, 8, 9, 10, 12, 15, 16\}$
$\{0, 1, 2, 3, 8, 10, 11, 12, 15, 16\}$
$\{0, 1, 2, 3, 8, 10, 12, 13, 15, 16\}$
$\{0, 1, 2, 3, 8, 10, 12, 14, 15, 16\}$

EBR 0.6 = {2,3,4,7,8,11,12,13,14,16}

Liver-disorders Database

a reduct = {1,2,4,6}

EBR 0.5 = {1,2,4,6}

Lung Cancer Database

some reducts = {0,8,13,26,56}

$\{0, 8, 13, 42, 56\}$

EBR 0.072727 = {0,8,13,44,56}

Pima Indians Diabetes Database (small)

(reduced to the first 300 instances) only reduct = {1,6,8}

EBR 0.3 = {5,6,8}

EBR 0.4 = {1,5,6,8}

Teaching Assistant Evaluation

a reduct = {1,2,4,5}

EBR 0.6 = {1,2,4,5}

Mood Dataset

This is the same dataset as used in this thesis to illustrate ID3/Entropy.

only reduct = {0,2,3,4}

EBR 0.8 = {0,1,3,4}

(ID3 = {0,2,3,4})

Fruit Dataset

only reduct = {0,1,2,4}

EBR 0.8 = {0,1,3,4}

(ID3 = {0,1,2,4})

A.1 Conclusion

For small datasets, EBR produces very similar (and in some cases identical) reducts to RSDR. However, only small datasets have been investigated here, further investigation into larger dataset reduction is required.