13

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF INFORMATION STUDIES FOR AFRICA

TEXT RETRIEVAL USING SELF-ORGANISED DOCUMENT MAP: THE CASE OF ILRI DIGITAL LIBRARY

A thesis submitted in partial fulfilment of the requirement for the degree of Master of Science in Information Science

BY
Mulugeta Bayeh Taye

June 2002

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF INFORMATION STUDIES FOR AFRICA

TEXT RETRIEVAL USING SELF-ORGANISED DOCUMENT MAP: THE CASE OF ILRI DIGITAL LIBRARY

By
MULUGETA BAYEH

Name and Signature of Members of the Examining Board

Ato Getachew Jemaneh, Chairman, Examining Board

Ato Dereje Teferri, Advisor

W/t Saba Amsalu, Advisor

Dr. Osei Adjei, External Examiner

Acknowledgements

I wish to express my deepest gratitude to my thesis advisors Ato Dereje Teferi and Saba Amsalu without whom this work would not have been possible. My greatest thanks also goes to Ato Tesfaye Biru, who apart from bringing the research area to may attention had the patience to review the various drafts despite the so many calls on his time. Their support has covered an overwhelming expertise in the technical and methodological issues as well as in the philosophical underpinnings of the field. I also wish to extend my sincerest gratitude to the support I have received from my family, colleagues at the International Livestock Research Institute and friends who have tirelessly attempted to create a more stimulating and effective environment for this work.

I wish to thank all at SISA for making some of the miseries less painful. I am also very grateful to my wife Linda Solomon whose continued care, tolerance and attention from far off made a whole lot of difference in this research and I know I would never be able to pay back. Thank you Muna baby and thank you all for the so many personal encouragements and for making the past two years more rewarding than I have ever imagined.

DEDICATED

TO

Linda (Muna) Solomon

DEDICATED

TO

Linda (Muna) Solomon

Table of contents

List of table

List of figures

List of appendices

Abstract

The current availability of large collections of full-text documents in electronic form emphasises the need for intelligent information retrieval techniques. Especially in the rapidly growing digital libraries and distributed access, it is important to have automatic methods for exploring document collections. In this study, the WEBSOM method is used with a quarter of century of research publications maintained by the International Livestock Research Institute for this task. The Self-Organising Map (SOM), also known as Kohonen's feature map (a means for automatically arranging high-dimensional statistical data), is used to position encoded documents onto a map that provides a general view into the text collection. The general view visualises similarity relations between the documents on a two-dimensional map display, which can be utilised in exploring the material rather than having to rely on traditional search expressions. Similar documents become mapped close to each other providing an intuitive mechanism and ease of access for maximising the institute's digital information and knowledge resources particularly for users with limited domain knowledge.

This study also sheds some light on the power of the SOM in solving problems of high-dimensional data. The trained SOM and the user interface are now usable to both browse the collection and to automatically map new documents. It can successfully make a distinction between the various types of documents and efficiently clusters similar publications to near by locations. It is quite evident that the WEBSOM can effectively visualize the results and is thus especially suitable for exploration tasks without the need to come up with search expressions, which may be difficult even with a rather clear idea of the desired information. The method is a major breakthrough with respect to the much harder problem, for which search methods are usually not even expected to offer much support, encountered when there exists only a vague idea of the object of interest. The same hold true if and when the area of interest resides at the outer edges of one's current knowledge.

This full-fledged report presents most of the situations that may be encountered in a project that explores the practical application of a WEBSOM method to solve the basic

problem of devising a suitable search expression, which could neither leave out relevant documents, nor produce long listings of irrelevant hits. The report also provides the general context of text retrieval and a detailed discussion on the actual method used in this research in the various sections. The step-by-step procedures and functions used in both encoding the document collection (preprocessing), computation of the Kohonen feature map and the development of the web-based map interface as well as a discussion of the essential results together with the codes used are included in the report.

Chapter One: Introduction

1.1 Background

Way back in 1950 when Calvin Mooers introduced the term 'information retrieval' into the literature of documentation, for better or worse the name stuck, notwithstanding the question it raises of what an 'information retriever' might be (Swanson 1988). Although, traditional query-based retrieval systems have since established themselves fairly well, recent exciting logical and technological advances continue to lead to more drastic changes in the way information is organised and retrieved (Kemp 1988). Cheap mass storage (using optical techniques particularly CD ROM), for instance, is providing for significant changes in the amount of data that can be retained for future use. Startling advances in computer networks also afford instant global access (the Internet for example) to tremendous diversity and volume of information making information overload an increasingly pressing research problem (Chen et al. 1998). Again, these advances are helping create a situation where virtually anyone or any institution with access to information technology tools can nowadays easily and with relatively little cost become an information provider for an unlimited audience (Lagus 1998).

Although these advances might not be said without sounding trait, the implications of them all continue to call for a shift from a simple query-based (key-word based search) to content-based search (based on the content similarity or relationship between documents) paradigm (Chen et al. 1998). This shift in the human-computer interaction paradigm is fuelled by the realisation that the same idea or concept can be presented in many different ways (Kaski et al. 1998). Natural language (the language in every day use) gives freedom for enormous variation in expression, from choosing between synonyms to using different styles, emphasis, different levels of abstraction, and metaphoric expressions. Furthermore, the authors use their unique style that depends on their background, knowledge and personal

style of communication. These variations in natural language potentially render information retrieval systems, based only on the keywords as they appear in the text, misleading because these methods have limited possibilities to tackle these phenomenon (Kaski et al. 1998; Lagus et al. 1998, Kohonen et al. 2000). These methods of organising and managing text have thus become both inadequate and too expensive to perform and to maintain for the majority of the available collection (Lagus 2000b).

Large quantities of textual data available for example on the Internet pose a continuing challenge to applications (like search engines) that help users in making sense of the data. Devising suitable search expressions for information retrieval from a full-text online databases may currently require considerable efforts (Kohonen et al. 2000). They often times heavily rely on the users competence to generate queries as much as the computer's ability to match the queries with information stored in the computer. Changing the scope of the search when, for example too many hits have been obtained, requires re-formulation of the search expression. At best, when interaction is involved, users can modify queries based on the retrieved results of a previous query. But again, this is analogous to searching books in a library without light by walking around from stack to stack in the dark, without knowing what stacks were walked through (Lin 1997; Lagus et al. 1996a). In this situation, success in finding a book greatly depends on whether one can walk to the right place in the dark (to generate a good query), and whether one knows how to adjust one's locations until one gets to the right place (to modify queries interactively).

Evidently, the problem is further complicated by the degree of similarity between and among documents. Most users have difficulty specifying their needs by a specific query formulation; even if users are successful in doing so, systems have difficulty retrieving all relevant documents without overwhelming the users with irrelevant documents (Lagus et al. 1996b; Lin 1997; Chen et al. 1998). A problem also arises when the exposure to the system and domain knowledge of the user are limited. Fulfilling a vague information need regarding an unknown domain, or obtaining an overview of a topic or a domain still remains more an illusion. Even these significant advances in information organisation and retrieval are left

much to be desired at when the answers sought relate to a set of documents instead of a single document, or when unexpected patterns or trends should be identified (Lagus et al. 1999; Lagus 2000b).

The other problem with these traditional query-based retrieval techniques is 'linear display' (Lin 1997). More often than not, documents matched to the query are displayed as a list of authors or a list of document titles. Even when relationships between and among those matching documents exist, the display is usually simplified and all the relationships are reduced to a one-dimensional, linear order, either chronologically, alphabetically, or according to some kind of relevance (weighted) order between the matching documents and the query.

In recent years, however, artificial neural network algorithms (like WEBSOM—Self Organising Map) have begun to be successfully applied to address a variety of these traditional query-based retrieval problems on complex document sets (Lagus and Kaski 1999). The method has been well-studied and developed especially for small-scale problems, although with increasing computing power it has also become possible to tackle much larger problems of document retrieval (Kohonen et al. 2000). Preliminary results in a text retrieval experiment indicate that the WEBSOM method and the resultant document maps (the medium for portraying the information collection) perform at least comparably with more conventional retrieval methods (Lagus 2000a). It has also been claimed that the method is scaleable and generally applicable on document collections of various sizes, text types, and languages (Lagus 2000b).

This research attempts to study the application of this popular artificial neural network algorithm to provide for the problem of document organisation and to ease exploration of livestock research documents maintained by the International Livestock Research Institute (ILRI), the leading global institute in livestock research. The motivation to the current research emerged from the personal experiences as a student on the organisation of ILIR's Publications Database and the potential application of WEBSOM to speedup text retrieval in a way that people can still browse and find the information they need.

The purpose of the study is to help reduce the complexity of information structures and facilitate users' access to and association and navigation of the text collection. It heeds the promise of the WEBSOM method (based on the Kohonen's feature map algorithm) 'to present information only at the most appropriate level of details to the user in order to avoid disorientation and cognitive overload' (Lin 1997; Wise 1999).

The self-organising map (SOM), developed by Teuvo Kohonen, is an unsupervised neural network algorithm applied to automatically organise large collections of text documents onto a two-dimensional display called the map (Kohonen et al. 2000; Kohonen 2001). Visual, map-like display provides a medium for portraying information about the collections of text. Relationships between text items and collections, such as similarity, clusters, gaps and outliers can be communicated using spatial relationships, shading, and colours. The method places documents on regularly spaced map grid points where similar documents are generally found near each other. With the help of HTML-based interactive tool the map can be used in browsing the document collection and in performing searches on a collection. The document landscapes also provide for visualisations of for example time-related properties of the data. Search results can be visualised in the context of related documents. Zoom operations can be used to focus on a detailed view of a sub-collection, and further zooming brings to view individual documents. Label words positioned on the map portray properties of the underlying map area. In addition, a search facility provides a means for describing a specific interesting topic and for finding a suitable starting point for exploration.

The current research primarily study the visualisation and user interaction aspect of the SOM algorithm. It examines the possible ways of utilising document maps to provide an intuitive user interface for accessing ILRI's textual document collection.

For the WEBSOM, the ILRI research publications database was used. These publications were the result of a quarter of a century of research in livestock production and disease control by the institute and its collaborators. The text corpus used consisted of the titles and abstracts of research publications (journal articles, research reports, conference/workshop proceedings, working papers, annual reports) organised into some 96 pre-classified

publication categories. The average length of the titles and abstracts was 158 words (the size of an average paragraph). These titles and abstracts are assumed to summarise the main themes of the publications.

1.2 Statement of the problem

One of the 16 international institutes sponsored by the Consultative Group (CG) on International Agricultural Research (CGIAR), ILRI has produced a wealth of information and knowledge that require efficient and automatic methods of organisation and interactive browsing to all type of users. With increased popularity of online digital libraries, the institute is also vigorously going back and digitising over a quarter of century of its research publications further justifying the need for such method if these resources are to continue to serve some purpose to the wide range of researchers within the national agricultural research systems and other sister institutions within the CG.

The increase in the sheer volume (the number of documents in digital form doubled from a few hundred documents only two years ago) of the collection coupled with the apparent similarity in the nature of the institute's collections (often limited to livestock production and disease control) is also continually outgrowing the current query-based system. It would not be late before users of the institute's information resources find it more and more difficult in selecting (locating) relevant documents from the mass of information available.

Again with the growth of digital libraries, it is also possible to have almost non-stop research reports streaming in. The reports need to be categorised so that they may, for example, be routed to those who declare an interest in a set of (related) categories. Automatic categorisation of research reports is of substantial interest to partner institutions, donor agencies, institutional strategy setting, researchers, information retrieval communities (application of technique with operational constraints) and to major partnering institutions (who would want to collaborate).

In addition, the situation is further complicated by the decentralised nature of the system that allows users direct access to the electronic text collection from their desktops. Even where intermediaries are at the system user's disposal, the psychological and behavioural problems that manifest particularly in cases of systems that serve researchers, academicians and other learned user groups poses another dimension to the problem. It is not uncommon for such users to be reluctant to accept support from intermediaries in the construction and optimisation of existing search facilities of the system (Dawit 1998). Devising suitable search expressions with the current system also depends on the users competence to come up with the right keyword. Both system users and providers share the view that the current system demands too much effort on the part of the users, which does not seem to go along with the whole purpose of investing a huge sum of money organising the collection. The time spent searching for relevant information is thus increasing beyond what most of the system users can sometimes justify. Equally important, the current system does also not provide for the apparent gaps in the research domain for young research affiliates of the institute who constitute a good number of the users of the system.

ILRI has also currently produced various computer based information systems. Some of these CD ROM-based virtual libraries include *ILRI on a disc* (containing years of the current institute's and its forbears' (ILCA and ILRAD) publications), *Virtual-SLP* (a CG System-wide Programme on livestock research), training materials on the worlds *Animal Genetic Resources*, etc. in addition to its long time presence on the web. The institute's efforts to make these resources public domain products by providing access in various forms could also be complemented by this work.

A discussion of the various advances in information retrieval methodologies is beyond the scope of this chapter. It suffice to say that in the past five decades various techniques and tools have been suggested ranging from how documents are represented inside the computer to how they are retrieved. One such recent methodological advance is the WEBSOM (details are provided in Chapter 3), a tool especially used in exploring a document collection as well as in searching and filtering tasks (Han and Kamber 2001). The method provides the

potential of tackling the problems associated with the query-based retrieval methods (discussed in the background) by presenting a map of the document collection, a map where documents are ordered according to their content, then even partial knowledge of the connections of the desired information to something already familiar would be useful (Kaski et al. 1998; Kohonen et al. 2000; Lagus 2000a). The map avails itself to visualisation, and thus the distance relations between different data items can be illustrated in a familiar and intuitive manner (Honkela 1997). This research therefore aims to provide for the problem of information organisation and improved retrieval of a document collection by applying Kohonen's feature map algorithm on ILRI Research Publications' Database.

In recent years, Kohonen, Lin, Lagus and others have been involved to try out and utilise this method to explore the possibilities of developing a retrieval system for searching large quantities of text information. Increased capabilities of computer technology for the manipulation of all types of information is also providing additional impetus to these research (Lagus 2000b). This work therefore is motivated by the need to provide an alternative and intelligent information retrieval scheme for text retrieval by applying the WEBSOM method. It attempts to address practical problems of speedy and interactive map-based access to users (particularly those with limited knowledge of the collection space) the institute's document collection.

The current retrieval system that has a predefined and fixed format, offering limited access by means of indexes, provides an ideal testbed for the hypothesis that a conventional query-based retrieval system may be transformed into an intuitive two-grid map to a document collection to improve retrieval. On these grounds, the current research attempts to investigate the applicability of the SOM algorithm to solve the problems associated with the present conventional retrieval environment.

The study will also, among others, attempt to look into the following retrieval challenges:

- How does recent advances in map-based organisation of document collection be used to address problems of information retrieval at the institute and help identify gaps in livestock research?

- Can recent logical advances in WEBSOM help facilitate the institute's effort to collect, organise and store as well as provide users of its information system better chances to explore and retrieve text documents? How can WEBSOM be used to make efficient utilisation of rare pieces of valuable texts to save the institution from continuing to be faced with a vast amount of textual documents of unknown value?
- Can WEBSOM reduce the number of items examined by all users of the information system at the institute to a manageable size?
- Can the SOM provide for the blind fold search (as some would like to call it) that characterises query-based retrieval systems of the institute? Can WEBSOM offer the promise of a lifetime for the institution to effectively manage its free-text based information resources?
- Will some visual cues (like a road map of a document collection) in our retrieval systems help our perception for information seeking in the digital environment?

1.3 Justification of the study

Theoretically, one may certainly concur that the availability of modern information retrieval techniques have greatly improved access to many stored information collections. However, a situation where it may not be possible to isolate the 'useful' items from the mass of available information is not far fetched yet because the current traditional query-based techniques are unable to deal effectively with the increasing growth of information. The existing information problems appear difficult to master without the use of more sophisticated methodologies for storing, processing and most importantly exploring (Lagus 2000b). Moreover, for the stored information to be useful the information must be displayed in a manner that can ease access to potential users (Lin 1997) because the goal of such a system is to aid the user in fulfilling his/her information need.

The visual text exploration, especially the map metaphor, is very recent. As a result, neither the possibilities nor the difficulties in the navigation of vast text collections using

visual landscapes have been fully explored (Lagus 2000b). In addition to helping solve an apparent problem of the institute therefore, this research might as well contribute to the ongoing global effort to test the scalability and generalisability of the WEBSOM method by comparing with the conventional query-based retrieval methods in a real world situation.

Besides, the map display may be implemented as a front-end interface to the ILRI online databases. Such an interface is believed to increase the number of documents that the user is willing to browse and will suggest many terms to the user when he or she needs to modify the query. The map interface may also be implemented in personal information systems to help livestock researchers at the institute to maintain and organise a personal collection.

Although the work in this research is mainly motivated by the applicability of the developed methods to practical problems regarding information retrieval, it may also serve as an inspiration for others to investigate the potential applications of the algorithm in areas for example of natural language processing.

1.4 Objectives

General Objectives

This research, based on providing an alternative scheme for text retrieval, emerged from the need for intelligent information retrieval techniques for exploration of full-text document collections in electronic form. It is motivated by the applicability of the SOM algorithm to address practical problems of speedy and interactive map-based access to users (particularly those with limited knowledge of the collection space) the institute's document collection. In addition, this work attempts to contribute to the ongoing global effort of testing the scalability and generalisability of the WEBSOM method by comparing with the conventional query-based retrieval methods in a real world situation.

Specific Objectives

The study will among other things attempts to:

- Review the research context of the WEBSOM method, namely text information retrieval, with emphasis on the visual and exploratory aspects in the context of large document collection.
- Investigate further the potential applicability of the WEBSOM in specific type of text document collection visualisation, exploration, and searching.
- Discuss other SOM-based studies to help put the current research into perspective as well as compare the results of this study with previous experiments. An attempt will also be made to compare and contrast the efficiency of a spatial presentation of and interaction with textual information to the traditional search-oriented approach by taking sample users at the institute.
- Discuss various challenges regarding visualisation of and navigation in a document landscape by testing a prototype of a system using the WEBSOM method.
- Test the potential of the demonstration interface for conveying and discussing additional ideas regarding map-based visualisation and navigation and traditional retrieval methods.
- Test the hypothesis that (with a standard test material) document maps can be used not only for exploration and visualisation, but also for successfully speeding up of information retrieval, and for improving retrieval results compared to standard methods.
- Explore the application of Kohonen's feature map algorithm to document visualisations by generating a map display.

1.5 Methodology

Review of related literature

The study involved review of related literature on the general context of text exploration in the context of large document collection and the actual method used in this research. Various sources including books, journal articles as well as literature from the Internet were

consulted. Other SOM-based studies were also surveyed to help put the current research into perspective as well as compare the results of this study with previous experiments.

The application of Kohonen's feature map algorithm to document visualisations is explored. The work also involved review of the appropriate indexing algorithm (to be used to 'survey' the contents of the document collection and 'detect' semantic relationships of terms and documents) and document encoding technique.

Data collection methods

To learn the structure and organisation of the research publication, various one-to-one and group discussion were held with the ILRI Information Services team responsible for managing the current publications database.

Programming techniques

This study used various programming techniques to implement the complex preprocessing (feature extraction for the purpose of document encoding) and user interface development phases of the work in addition to a SOM implementation software tool called Nenet. Visual Basic programming has been used for encoding the required functions and procedures to extract the key terms from both the titles and abstract that represent the document features and write them to document vectors. A Porter stemming algorithm, a popular English word stemmer, is also integrated to the Visual Basic code that does the feature extraction.

The development of the browsing interface on the other hand required the development of appropriate HTML pages (with the few lines of Java Script) and implementation of ASP codes (VBScript). The ASP (Active Server Pages) code implements the actual fetching function that helps to access the database while the few lines of Java Script integrated into the HTML code read the cursor position (in terms of the x and y coordinates) of the map image used in the exploration of the document collection. The codes used for both phases (the document encoding and development of the user interface) are annexed for reference.

Note that the document collection is organised into an MS Access database that includes various tables for the training and test datasets as well as the various other tables that were created and maintained to keep track of records used in the many functions and procedures at the different stages of the document feature extraction.

Testing techniques

A set of standard test procedures verified both in the WEBSOM literature and Nenet documentation were undertaken to measure the quality of the mapping and test whether the map has adapted to the input data vector. Three such tests, using different data vectors prepared to measure the efficiency of ordering and clustering of the trained map, were separately conducted.

The efficiency of the spatial presentation of and interaction with textual information to the traditional search-oriented approach has also been tested with a few users of the collection. An attempt has also been made to compare the retrieval results of the web-based visual interface with the retrieval results of the current traditional keyword based search.

1.6 Scope and limitation of the study

The scope of this study is therefore limited the development of an automatic method for information retrieval for users with limited knowledge of the institute's collection. Within the framework of the WEBSOM architecture, various tasks have been undertaken to accomplish this. These include developing the appropriate functions and procedures to help extract the document features and encode the document collection into data vectors, selecting the appropriate parameters to initialise and train the Kohonen feature map, testing the efficiency of ordering and clustering the trained map and the development of the user interface. For computational reason, the experiment is limited to the titles and abstracts of the research publications. The document collection contains some 987 documents organised in an InMagic full-text database.

The work of this thesis attempts to study the possibilities and the difficulties in the navigation of vast text collections using visual landscapes, especially the map metaphor. In addition to helping solve an apparent problem of the institute therefore, this research is meant to contribute to the ongoing global effort to test the scalability and generalisability of the WEBSOM method by comparing with the conventional query-based retrieval methods in a real world situation.

The only major limitations to these study was time and lack of detail in the WEBSOM literature. While shortage of time left no chance to compare the results of this study with other approaches and techniques, the lack of details in the WEBSOM literature made implementation of some of the advanced features impossible. Paradoxically, however, the major strengths of this report emanated from this gap in details in the WEBSOM literature and attempt has been made to cover enough ground to avoid any such pitfall in the future.

1.7 Structure of the thesis

This thesis is structured into five chapters. It provides an introductory chapter (this section) with an introduction, statement of the problem and objectives of the current work. Chapter two reviews related literature on the general context of text exploration in the context of large document collection. It particularly discusses the issue of comparing the different means of representation of a query with representation of the texts for the purpose of retrieval. Together with Chapter 3, that discusses the actual method (WEBSOM) used in this project, they provide the context for the thesis by describing the different techniques for exploration and some aspects of the generation of the representation. A section on other SOM-based studies have also been included to help put the current research into perspective as well as compare the results of this study with previous experiments.

Chapter four presents the actual application of Kohonen's feature map algorithm to ILRI document collection and the development of the user interface as well as discussion of the result. The chapter also provides for a small note on the efficiency of a spatial presentation of

and interaction with textual information to the traditional search-oriented approach by taking sample users at the institute. The last chapter (Chapter five) summaries the current work and attempts to indicate potential areas of follow up research related to this work.

Chapter Two: Text mining

Where is the wisdom we have lost in knowledge? Where is the knowledge that we have lost in information? T.S. Eliot, The Rock (1934)

2.1 Text mining

Advances in information and communication technology has significantly revolutionised our possibilities to collect, generate, distribute and store text data beyond what any one magic-bullet system can afford to provide a solution to determine its value (Lagus 2000a). Text databases are thus rapidly growing due to the increasing amount of information available in electronic format, such as electronic publications, email, CDROM and World Wide Web (WWW) (which can also viewed as a huge, interconnected, dynamic text database). The need to process this large, growing collections of electronic documents which consists of large collection of documents from various sources such as journal articles, research papers, books, digital libraries has become everyday challenge.

The situation is further complicated by how the text data can be structured in storage. Data stored in most text databases are only semi-structured in that they are neither completely unstructured nor completely structured (Han and Kamber 2001). For example, a

document may contain a few structured fields, such as titles, authors, publications date, length, categories and so on, but also contains some largely unstructured text components, such as abstracts, contents and full text.

In addition, the ever-increasing number of institutions (including the International Livestock Research Institute (ILRI) whose document collection this research project considers) with access to these technologies growing to become information providers for an unlimited audience (see previous Chapter for details of problem description) poses another dimension to the problem. As a result, instead of rare piece of valuable texts, these institutions and all the people they cater for are increasingly faced with a vast amount of digital information of less known value. The use of automatic methods, algorithms, and tools for dealing with large amounts of data, especially of textual nature, has become necessary (Honkela 1996; Han and Kamber 2001). Attempts to solve this general problem of locating relevant documents based on user inputs can be loosely described as text mining (Lagus 2000b). Text mining can also be viewed as a specific field of data mining[1] (Han and Kamber 2001).

This section attempts to describe a specific aspect of research and development in the oldest and most established field of text mining (Smith 1987)—text retrieval systems—i.e. the means for identifying, retrieving and/or ranking texts (or text surrogates or portions of texts), in a collection of texts, that might be relevant to a given query[2] (or useful for resolving a particular problem). In particular, it discusses the issue of comparing a representation of a query with representation of the texts for the purpose of retrieval. Together with other sections in this chapter (that discusses the different means of representation), it provides the context for the actual research (SOM-based text retrieval) by describing the different techniques for exploration and some aspects of the generation of the representation.

Figure 2.1 presents the situation with which this research is concerned.

[1] Data mining is the analysis of (often large) structured data sets to find unsuspected relationships and to summarise the data in novel ways which are both understandable and useful to the database owner (Han and Kamber 2001).
[2] Set of words/terms fourmulated by the user.

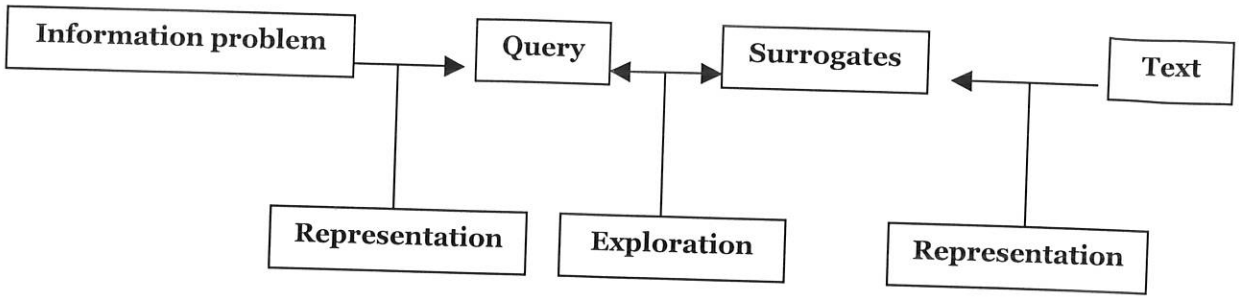


Figure 2.1. *A simple text retrieval scenario.*

Source: Adapted from Belkin and Croft (1987).

It is important to note from the outset that the number of techniques used for retrieval, or covered in the IR literature as having to do with text retrieval, will be discussed only as aspects of simpler, or basic matching techniques. In the context of Figure 2.1 above, for instance, feedback techniques will be treated rather like a method for enhancing the query than a technique on its own.

With this brief background note on context of this project, the rest of the section is structured as follows. It begins with a brief overview of the concept of information needs and tasks related to texts. It then present a classification of retrieval techniques and use this classification as the basis for discussing specific research and development in retrieval techniques. Comparative studies and the relationship of request representation in retrieval techniques are discussed. As part of the effort to put this exercise in perspective the chapter wind up with a general discussion of the requirements for text retrieval systems and description of research directions that seem to be the most pressing in developing truly effective text retrieval techniques.

2.2 Information needs and tasks related to texts

In a nutshell, the goal of a retrieval system is to aid the user in fulfilling his/her information need. In some cases, a specific question needs to be answered, or certain document to be found, whereas in other cases an overview of a topic is desired. At other times, the need is just to merely find 'something interesting', or to obtain a general understanding of 'what is

out there', or to find unexpected patterns or other new information. Furthermore, the need may be only vaguely understood by the user, and in some cases difficult to express in natural language (Lin 1997).

Chen et al. (1998) and Lagus (2000b) describe three major tasks related to various information needs: searching, browsing and visualisation.

Searching: In searching the user specifies his/her need by a query and expects the system to locate individual documents that correspond to it. Internet search engines are a familiar example of such tools. This approach presupposes a clear knowledge of the user what is to be found, and a potential to explicitly express them. However, the need may more often be vague, the domain unknown, and the user is constrained by the appropriate, specialised vocabulary (often called the vocabulary problem).

Browsing: In browsing, the user navigates via links between individual documents (WWW for instance) or via some hierarchical structure such as the contents section of a book or the directory structure of an explorer window. The browsing approach reduces the information need to be more vague or unconscious, since no explicit description of the need is required (Chen et al. 1998; Lagus 2000b). Instead, the need is implicitly communicated via the choices made in browsing, such as the links followed.

In both searching and browsing the background assumption is that the user can be satisfied by individual documents. However, when the need is either very vague, or very general, providing access to even the most appropriate individual documents might not fulfil the need (Lagus 2000b). In such cases, summary-like information might be more appropriate and useful, for instance.

Visualisation: In visualisation of information something familiar is used as a means for illustrating something unfamiliar. As pointed out, there exists information needs that require assessing and conveying similarities, differences, overlaps, and other relationships between collection of documents. One might, for example, wish to find out the relationship between a familiar set of documents, e.g. personal files or familiar mailing list, to yet unfamiliar

collection. Using suitable visualisations intricate relationships between large collections of items can be communicated fast and intuitively.

Shneiderman (Quoted by Lagus 2000b) identifies the following seven major tasks that a visual and interactive information exploration system provides:

- Gain an overview of the entire collection
- Zoom in on items of interest
- Filter out uninteresting items
- Select an item or group and get details when needed
- View relationships among items
- Keep a history of actions to support undo, redo and progressive refinement
- Allow extraction of sub-collections and the study of their properties.

The most useful tools for text retrieval will in the future encompass all the above aspects (Belkin and Croft 1987; Chen et al. 1998; Lagus 2000b), providing a variety of means to explore large collections of text by enabling alteration between visualisation, browsing and searching.

2.3 Information retrieval

In its simplest form, an information retrieval (IR) system can be viewed as 'the representation, storage, organisation, and access of information items' (Salton and McGill 1987). The general field of text retrieval can be characterised as follows (Belkin and Croft 1987; Harter and Hert 1997; Lagus 2000b; Han and Kamber 2001). Traditional IR techniques (see section 2.2.1) become inadequate for increasingly vast amounts of text data. There are a few techniques actually used in large-scale operational IR systems (namely, Boolean or string searching), and there appears to be some general dissatisfaction with them at both the inadequacy of performance and cost of maintenance. These techniques have become established more through practice than theory. At the same time, there are a number of quite different techniques (see section 2.2.1), usually with a strong theoretical basis, that

have been used almost exclusively in experimental settings (e.g., probabilistic retrieval). However, these are now well developed in many of their aspects. These 'experimental' techniques, when compared with the 'operational' techniques in a controlled setting, have almost always performed better on standard measures (recall[3] and precession[4]) and often very much better. Although these two types of techniques (operational and experimental) are now well established in their respective settings, the experimental experience had so little effect on the operational environment. This research could be partly described as an effort in field-testing the potential of one such technique called WEBSOM.

For some time a number of researchers in the field have noted that all retrieval techniques perform better for some queries than for others. In cases where techniques have been compared with one another at a micro level, it appears that differences in performance on specific queries are masked by evaluation of cumulated results, and that especially, even when specific techniques have been shown to perform more poorly than another technique overall, they may have done much better on at least some individual queries. Thus, are some retrieval techniques better for some kinds of queries (generated with limited domain knowledge of the collection, for instance) than others? There is sufficient justification now to think that such is the case which is what WEBSOM is all about.

The research in IR focuses on the retrieval problem (locate relevant documents based on user input, such as keywords or example documents) from a vast amounts of information to which accurate and speedy access is becoming ever more difficult, i.e., on the situation where it is 'assumed' that the information need can be explicitly and adequately described. This however not often the case further justifying the need for an alternative framework of presenting information without compromising on the speed of access and still locating relevant information.

---

[3] The percentage of documents that are relevant to the query and were, in fact retrieved.
[4] The percentage of retrieved documents that are in fact relevant to the query (i.e., correct response).

2.3.1 Retrieval techniques

This review of retrieval techniques and the associated problems in IR systems is based on Belkin and Croft (1987) classification that establishes meaningful relationships among the classes of objects involved (see Figure 2.1) than an historical accident of custom (setting) or use (experimental vs. operational).

Retrieval techniques[5] can be described in terms of the characteristics of the retrieved set of documents (exact vs. partial match) and the representations (on their own or as part of a whole) that are used. Although some techniques do not fall naturally into only a single category in this classification, and others are hybrids of techniques from different categories, but the scheme is useful for discussing the broad distinction among retrieval techniques. Figure 2.2 (adapted from Belkin and Croft 1987) gives a diagramatic view of the classification scheme discussed in this thesis.

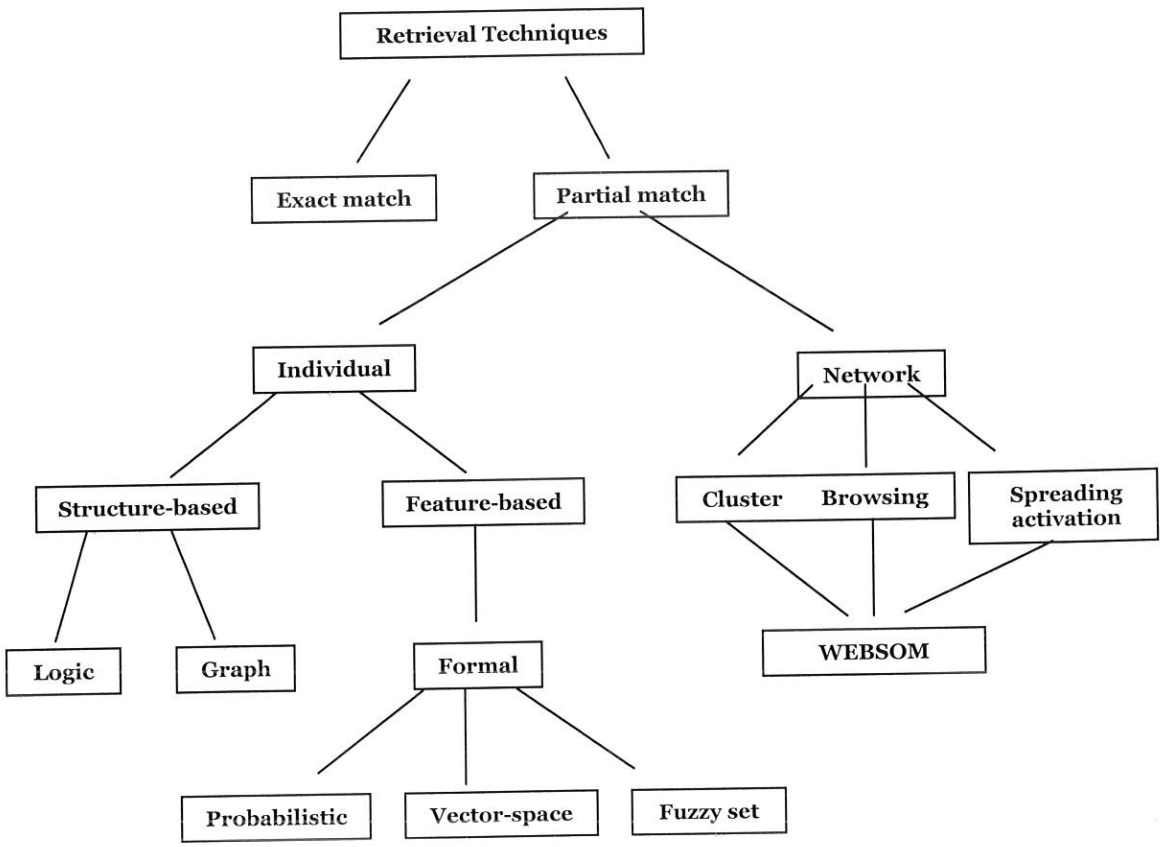


Figure 2.2. *A classification of retrieval techniques (Adapted from Belkin and Croft 1987).*

The first distinction made is whether the set of retrieved documents contains only documents whose representations are an exact match with the query or a partial match with the query. For a partial match, the set of retrieved documents will include also those that are an exact match with the query.

The next level of the classification distinguishes between retrieval techniques that compare the query with individual document representatives and techniques that use a representation of documents that emphasises connections to other documents in a network. In this category, individual documents are retrieved, but the retrieval is based on connections to other documents and not solely on the contents of an individual document. In the network category, we identify the subcategories of cluster-based searches, searches based on browsing a network of documents, and spreading activation searches, that together seem to dictate the future direction of IR research (Belkin and Croft 1987).

The individual category breaks down into retrieval techniques that use a feature-based representation of queries and documents and techniques that use a structure-based representation. In a feature-based representation, queries and documents are represented as sets of features, such as index terms. Features can be weighted and can represent more complex entities in the text than single words. The structure-based category is divided into representations based on logic, that is, those in which the meaning of queries and documents are represented using some formal logic, and on representations that are similar to graphs, in which documents and queries are represented by graph-like structures composed of nodes and edges connecting these nodes.

The feature-based category includes techniques based on formal models: vector space model, probabilistic model, fuzzy set model and others) (see discussion on each of these models below). The following section discusses the various techniques that make up these categories in more detail. Note however that a complete review of all the techniques and models would not only be futile but also out side the scope of this project. Reference to

[5] A retrieval technique can be defined as a technique for comparing the query with the document representation (Belkin and Croft 1987)

sources for a detailed discussion on the various attempts to alternative frameworks is instead provided as deemed appropriate.

A Exact match techniques

Exact match retrieval techniques are those that require that the query be precisely contained within the text representation. Implemented as Boolean, full-text, or string searching, this is the retrieval technique in use in most of the large operational IR systems (Belkin and Croft 1987). The disadvantages of this type of technique are well known and well documented and a variety of aids such as thesauri is used to improve retrieval performance. In the simple case exact match searching: 1) misses many relevant texts whose representations match the query only partially; 2) does not rank retrieved texts; 3) cannot take into account the relative importance of concepts either within the query or within the text; 4) requires complicated query logic formulation; and depends on the two representations being compared having been drawn from the same vocabulary. Honkela (1997) mentions several other undesirable characteristics of Boolean retrieval.

Despite the many objections, the exact match searching still remain the paradigm for operational systems (Han and Kamber 2001). The traditional justifications are that the investment in current systems is so great that changing them is economically unfeasible, that alternative techniques are untested in large-scale environments, and that the results of alternative techniques are not sufficiently better even in experimental environments to justify any changes (Harter and Hert 1997). A more significant argument is that the structures of Boolean statements represent important aspects of user's queries or problems (Belkin and Croft 1987).

Research in exact matching retrieval techniques has dealt with all the problems mentioned above to some extent and with one additional problem: that of efficiency of searching for strings (Belkin and Croft 1987; Honkela 1997). The major efforts in the logic of exact match retrieval have been in making less exact (see the section on partial matching), in taking into account relative importance, and in achieving sensible ranking rules.

B Partial matching techniques

Feature-based techniques

Retrieval techniques in this category are used to compare queries with documents represented as sets of features or index terms. The document representatives are derived from the text of the document either by manual or automated methods. Similarly, the query terms are either derived from a query expressed in natural language, or an indexing vocabulary is used directly for specifying queries.

Features represent single words, stems, phrases, or concepts and can have weights associated with them. The weights are typically derived from the way the term is used in an individual document, such as a within-document frequency weight[6], or the way it is used in the document collection, such as the inverse document frequency weight[7].

In the following discussion of feature-based retrieval techniques, one assumes that a document has a representation consisting of a vector of terms (d_1, d_2, ... d_m), where d_i indicates the presence or absence of term *i* and has the value 1 or 0. Weights associated with these terms are introduced as needed. A query has a similar representation, with q_i referring to the *i*th query term. Alternative methods to the purely feature-based approach but are strongly related to it, such as the partial match Boolean query techniques, are also given a passing remark.

Formal. These retrieval techniques are based on formal models of document retrieval and indexing. The major modelling approaches that have been used for information retrieval are vector space, probabilistic, and fuzzy set (Belkin and Croft 1987). Further discussions of these models of information retrieval are discussed by Salton and McGill (1983) and Van Rijsbergen (1979).

Vector space model: In the vector space model, documents and queries are vectors in an n-dimensional space, where each dimension corresponds to an index term. Proposed by Salton, the model has formed the basis of a large part of IR research (Honkela 1997). One explanation for this is that vector operations can be performed very fast, and efficient

[6] The number of times a term appears in a document.
[7] The inverse of the number of documents in which a term occurs.

standard algorithms exist to manipulate vectors (Lagus 2000b). Although this was one of the first models proposed, modifications to it are still appearing (Belkin and Croft 1987; Honkela 1997). The model works in the following three steps:

1. Calculate term weights by a combination of the within-document frequency (tf) and the inverse document frequency (idf).
2. Replace terms that have poor discrimination value[8] by low-frequency terms and phrases for high-frequency terms.
3. Rank documents in decreasing order of similarity to the query as measured by some metric distance retrieving those documents closest to the query in vector space. This could be calculated by following formula:

$$\frac{\sum d_i q_i}{\sqrt{\left(d_i^{\,2}\, q_i^{\,2}\right)}}$$

where d_i is the tf * idf weight.

Belkin and Croft (1987) observe three important points about the vector space model that also apply to other partial matching retrieval techniques. First, the formula for calculating term weights, such as tf * idf, may vary from one system[9]. Second, when similarity measures are used, the query is not directly compared with every document in the collection to produce a ranking. Finally, although the term weighting done in steps 1 and 2 is often regarded as part of the indexing process, it can also be done during retrieval. These details can have significant impact on the effectiveness of a system.

An important extension of the vector space model based retrieval technique is extended Boolean retrieval (Belkin and Croft 1987). This technique overlaps the structure-based category because it uses structured (Boolean) queries. That is, the query is formulated with index terms and the Boolean operators AND, OR and NOT. In this approach, a similarity

[8] Terms that are not useful for distinguishing among documents. Discrimination values are calculated by observing the document space before and after assignment of a term. If the documents tend to move together after assignment (as measured by the average pairwise similarity), the term is a poor discriminator.

[9] For example, it is common in calculating the idf weights to normalise the document collection frequency with the maximum collection frequency rather than simply the number of documents in the collection. Like wise, tf weights are normalised with the maximum within-document frequency, and in others it is not normalised (Belkin and Croft 1987).

measure[10] is defined that ranks documents, giving precedence to those that match all or part of the Boolean specification. For example, consider a situation in which document 1 contains terms A and B, document 2 contains terms B and C and the query is A AND (B OR C). Ignoring the effect of term weights, the standard cosine correlation would rank documents 1 and 2 equally because both have two query terms. The extended Boolean similarity measure would rank document 1 higher because it matches the Boolean query specification (Note that the addition of term weights makes the calculation more complex, but the general effect is the same).

Research based on the vector space model has led to other techniques, such as clustering and document space modification (Belkin and Croft 1987). These techniques will be briefly introduced later in this chapter.

Probabilistic model: The retrieval techniques based on the probabilistic model are very similar to those developed from the vector space model (Belkin and Croft 1987). The basic aim is to retrieve documents in order of their probability of relevance to the query. If we assume that document term weights are either 1 or 0 and that terms are independent of each other, this can be achieved by ranking documents according to the summation of the product of d_i and q_i:

$$\sum d_i q_i$$

where q_i is a weight equal to log pr_i $(1-pnr_i)/pnr_i$ $(1-pr_i)$ where pr_i is the probability that term i occurs in the relevant set of documents, and pnr_i is the probability that term i occurs in the nonrelevant set of documents.

The problem in applying this ranking function is to estimate the probabilities in the query term weights (Belkin and Croft 1987).

A number of proposals have been made to remove the term independence assumption of the probabilistic model (Belkin and Croft 1987). These include identify important dependencies in the query and to use the presence of those dependencies to modify the document scores according to a probabilistic model that assumes term dependence. This

[10] A measure of the overlap of the query and document sets of terms.

avoids the calculation of dependencies that are not used in queries and identifies those dependencies most likely to affect retrieval effectiveness. The dependencies used by this technique can be identified using groups of query terms joined with the Boolean AND, thereby allowing structured queries to be used with a probabilistic retrieval model.

Fuzzy set: The basic idea of the fuzzy set technique to IR is that elements or entities can be assigned to sets to varying degrees, i.e. instead of inclusion or exclusion of an element to a set a membership function is used to express the degree to which the element is a member of a set. The assignment of individual words to meaning categories is a fuzzy process, so is the assignment of documents to concept categories and hence also the retrieval of documents in answer to certain queries (Salton and McGill 1983). The main contribution of this work in terms of retrieval techniques has been the integration of Boolean queries with ranking techniques (Belkin and Croft 1987). This integration is limited, however, when compared with extended Boolean retrieval based on the vector space model or the use of term dependencies in probabilistic models.

Structure-based techniques

In the individual, structure-based category of retrieval techniques, either the query or the documents or both are represented by more complicated structures than the sets of terms used in feature-based techniques. These techniques typically rely on a much richer representation of the knowledge in the subject domain covered by the documents and queries (Belkin and Croft 1987).

Logic: This approach to information retrieval is based on the theory that the information content of a text in documents can be represented as sentences in a formal logic. The statement, 'Dell sells computers,' for instance, could be represented in first-order predicate calculus as sells (dell, computer). Similarly, the statement, 'If a company invest in IT, it is financially viable,' could be represented as (forall (x) (if (invest x IT) (viable x))). Given a logic representation of document content, a query in the same logic could be answered by inference using the rules associated with that logic. For example, the query (viable?) can be

answered from the sentences given above. The critical problem with this approach to retrieval is the translation of the text into logic (Belkin and Croft 1987).

Graph: A number of structures fall into this general category. The general characteristic of a graph-like representation is a set of nodes and edges (or links) connecting these nodes. Retrieval techniques in this category look for similarities in the structures of query and document graphs. This similarity is then used to determine if a document should be retrieved or to modify a document ranking.

Graph-based techniques utilise the general method of first searching through the text collection, omitting graph matching of any sort, and then using the graphs to order the retrieved set. The reason for this is that graph matching is computationally difficult, and searching the entire database is always an intractable problem (Belkin and Croft 1987).

Network

Cluster: A cluster is a group of documents whose contents are similar. A structure-based retrieval technique, clustering works by a cluster hierarchy, using a clustering technique, and dividing documents into a few large clusters, dividing these clusters into smaller clusters, and so on. A top-down search of the cluster hierarchy is performed by comparing (using a similarity measure) the query to cluster representatives of the top-level (largest) clusters, choosing the best clusters, comparing the query with representatives of lower-level clusters within these clusters, and so on until a ranked list of lowest-level clusters is produced. The documents in the topranked clusters are then ranked individually for presentation to the user. A cluster representative can be generated in various ways, but in general it represents the average properties of documents in the cluster.

The cluster hypothesis was introduced as a basis for using cluster searches to improve retrieval effectiveness relative to ranking individual documents (Belkin and Croft 1987).

The emphasis on small, well-defined clusters has led to the development of retrieval techniques based on the generation of the document's nearest neighbours on which WESOM is primarily based (see detail in later section and Chapter 3). A document's nearest neighbours are those most similar to it, and a cluster of nearest neighbours is very similar to

the lowest-level single-link clusters. Given a network of documents connected to their nearest neighbours, it is possible to generate clusters and their representatives at search time with considerable storage savings (Belkin and Croft 1987).

Browsing: If the documents, terms, and other bibliographic information are represented in the system as a network of nodes and connections, the user can browse through this network with system assistance. Browsing is an interesting retrieval technique in that it places less emphasis on query formulation than do other techniques and relies heavily on the immediate feedback provided by user browsing decisions (Lin 1997). Through dialogue with the user, the system uses the network to build a model of the user's information need that includes relevant documents found during the process.

Other research in browsing (WEBSOM for instance) concentrates on the use of visual representations of the document database to acquire information from the user interactively.

Spreading activation: Spreading activation is a retrieval technique that has some similarities to browsing. A query is used to 'activate' parts of a network that describes the contents of documents and how they are related to each other. In the simplest case, the query would activate index term nodes that are connected to document nodes and other terms. The links and nodes represent concepts from the subject domain and how they relate to each other as well as the documents that contain those concepts. From the 'start nodes' provided by the query, other nodes connected to those nodes are in turn 'activated' (hence, the term 'spreading activation'). Criteria, such as threshold values that decrease as the activation propagates through the network or rules about the reasonableness of the inference implied by using a particular link, are used to control the spread of activation. Activation can converge on particular document nodes from a number of links. These highly activated nodes are retrieved.

In a simple network of documents and terms, the documents that have a high level of activation after the first links from the query nodes are followed will be those documents that have a high number of terms in common with the query. If the activation spreads to other terms connected to those documents and then to other documents, the documents retrieved

in this second phase will be similar to those found by a cluster search based on nearest neighbours. When the activation is refined using inference rules and more link types, it is more difficult to relate the retrieved documents to those found with conventional techniques (Belkin and Croft 1987). The retrieval technique in this case is more similar to structure-based techniques.

WEBSOM: In the WEBSOM method documents are organised using a self-organising map algorithm (Kohonen 2001) on to a document map. A graphical display of the map provides a general overview of the information contained in the document collection (Kaski et al. 1998). The map display can then be readily used to explore the document collection. They can also be labelled with automatically identified descriptive terms that convey properties of each area and acts as a landmark during exploration. With the help an HTML-base interactive tool the ordered landscape is then used in browsing the collection and in performing searches on the map.

The WEBSOM method can be described as a combination of the cluster, browsing and spread activation retrieval techniques discussed above. The method also addresses the vocabulary problem of previous techniques by organising the words into categories on a word category map (also called self-organising semantic map). The word category maps are SOMs, which have organised words according to similarities in their roles in different contexts. They can then be used to encode documents in a manner that explicitly express the similarity of the word meanings to take into account the fundamental problem of vector space model. Each unit on the SOM corresponds to a word category that consists of a set of similar words.

A detailed discussion of the WEBSOM method is presented in Chapter 3.

2.3.2 Retrieval performance

Various comparative performance studies to evaluate IR systems have been a major topic of research for a number of years (Van Rijsbergen 1979; Harter and Hert 1997). Although there have been and continue to be problems with the evaluations that have been done (Salton and

McGill 1983), these results provide valuable information about the relative performance of retrieval techniques.

Belkin and Croft (1987) observes the first important result is that all available evidence points to the superiority of partial match techniques over exact match techniques. They further note although there are problems with making direct comparisons between the sets of retrieved documents, it appears that the difference in effectiveness is significant. For feature-based retrieval, the evidence indicates that the best performance is provided by the probabilistic retrieval strategy incorporating term significance weights or its equivalent, the tf * idf/cosine correlation combination. This retrieval technique uses a simple similarity measure (the inner product) and the effectiveness is due entirely to the index term weights used. Results that indicate superiority of one technique over another in this context can be interpreted in terms of the estimates used for the weights. The use of good estimates is the major factor in obtaining the best performance from these techniques (Harter and Hert 1997).

2.3.3 Request model and retrieval techniques

It has probably become abundantly clear that it is difficult to separate the retrieval technique from the form of representation used in the request model. There is, however, a clear message from the performance on retrieval techniques. Whatever retrieval technique is used, the quality of the results depends almost entirely on the accuracy of the information in the request model (Belkin and Croft 1987). It is this process of an intermediary's interaction with the user to formulate the query that is the heart of current retrieval systems, and it is also crucial for more advanced retrieval systems. This fact has led to a lot of research in more interactive intermediary systems (e.g., Kohnon, Lagus, Lin). These systems engage the user in a dialogue with a variety of facilities to acquire a detailed request model.

More recently, a paradigm shift has begun in text retrieval from the search-oriented approach towards a special presentation of and interaction with textual information (Honkela 1997; Lin 1997; Lagus 2000b; Kohonen 2001). Properties of large sets of textual

items, e.g., words, concepts, topics or documents, can be visualised using one-, two- or three-dimensional space, or networks and trees of interconnected objects. Furthermore, this retrieval technique has allowed interactive operations, such as selection of a subset or a single item can be performed e.g. by pointing and clicking with the mouse.

There therefore is now some reason for optimism because, as this review has shown, not only is there dissatisfaction with current retrieval techniques, but significant work is at least beginning to attack most issues discussed. More research is being done to improve existing techniques and the relationship between technique and request model, on the testing of experimental techniques in operational environments (like what this project is all about), as well as on the specification and implementation of new architectures in text retrieval.

A successful IR system should therefore attempt to tackle the most important problem areas of information retrieval, in addition to helping support all the basic tasks related to information need: searching, exploration and browsing (filtering). Text retrieval system, Honkela (1997) also enumerates should:

- Require as little human intervention as possible in order to enable processing of very large document collections
- In searching, provide the results in the order of relevancy and offer means of exploiting what was left out from the result
- Support exploration by providing views on the document collection
- Degrade gracefully even if the documents are poorly written
- Be computationally feasible
- Take in to account, in a general way, the inherent features of natural language and adapt to the specific use of the system.

The network category, combined with the principles of the vector space model, appears to be a promising alternative for the traditional keyword-based methods (Honkela 1997). Chapter 3 presents a detailed discussion of this new SOM-based method for information retrieval called WEBSOM. The WEBSOM aims at satisfying the requirements listed above.

2.3.4 Organising documents with SOM

Several studies have been published on SOMs that map words or word forms they contain (e.g. Kohonen, Honkela, Lagus, Kaski, Lin). In an early study, a small map of scientific documents was formed based on the word forms in their titles (Lin 1997). Lin also later extended his method to full text documents. Scholtes and Honkela have applied (Lagus 2000a) the SOM extensively in natural language processing and information retrieval. In addition, to encoding the documents based on their words, Scholtes has used character n-grams in the encoding. SOM has also been used to cluster textual descriptions of software library components. Lagus et al. (1998; 2000) further developed the application of SOM by designing a browsing environment, which utilises the order of the document map so that the document collection may be explored with any graphical WWW browser.

A few attempts (by Kohonen, Lagus and others) have been made to test the WEBSOM method in realistic applications. Honkela et al. (1996) have performed a case study with material from Usenet new groups. Later (Lagus et al. 1997) used the method to organise scientific articles and 10,000 European patent abstracts in English, as well as news bulletins in Finnish. Demonstrations of some of these document maps can be explored at the WWW address http://websom.hut.fi/websom. The size of the document collections have varied from 60 documents to over a million documents organised on a single document map.

In these works these researchers have demonstrated how the new method could be used in organising collections of documents into maps. Since the first results, the WEBSOM method has undergone significant developments (Honkela 1997; Kohonen 1998). The developments include:

- Methods that enable computation of vary large maps have been developed (Kohonen 2001). The largest map so far contains over a million documents from patent abstracts.
- The quality of the large maps has been improved by enlarging the word category maps considerably. The overloading problem of word category maps, i.e., the problem caused by each map node containing too large a number of words, has been overcome.

- A new mode of using the document maps is based on content addressable search. The user provides a query, a list of terms or a document. The nodes that best represent the query are then highlighted on the map display to facilitate the exploration of the map.
- Recently, a method of automatically labelling of the document maps has been developed. For the exploration tasks, the labels provide information on the contents of the nodes. For small maps the labels can be added manually but when large maps are considered automatic labelling is needed.

2.3.5 Research directions in IR

In summary the results of research on IR techniques demonstrate fairly conclusively the inadequacy of the exact match paradigm for effective information retrieval and the status of operational IR systems, which use almost exclusively just one exact match technique—Boolean logic. This, of course, is not exactly new. What is perhaps new is the directions in which IR technique research is going. In the past, most research in retrieval techniques was rather far removed from operational environments and to some extent even from operational constraints. The past several years seem to show movement in this research in three directions (Belkin and Croft 1987; Harter and Hert 1997):

First there has been a good deal of work on relating partial match techniques to exact matching, as in extended Boolean searching and the use of Boolean-derived dependencies in probabilistic searching. Similarly, there are at least a few studies being carried out of partial-match techniques in operational environments (e.g., Kohonen, Lagus), which are explicitly designed to attend to the criticism that such techniques have never been demonstrated to be worthwhile in large systems. There are also several microcomputer-based systems that use partial match techniques in operational, albeit smaller, environments e.g., the WEBSOM (Kohonen)

The second response seems to be the realisation that no one technique will be adequate for all purposes and that either a mix of techniques or a principled choice of techniques is required to improve IR system performance.

The third new direction is increasingly complex representations of the request or user's problem. As noted, although retrieval techniques are not the same as representations, the techniques one can use are determined by the representation. The more complex the representation the more kinds of retrieval techniques are possible. Much of the work of this type has had its roots in the knowledge representation schemes associated with artificial intelligence and neural network research. In general, this work has arisen because of the need for an increased understanding of the importance of the request model.

Chapter three: The WEBSOM method

Languages are in some respects like maps. If each of us sees the world from our particular perspective, then an individual's language is in a sense, like a map of their world. Trying to understand another person is like trying to read a map, their map, a map of the world from their perspective. **Moore and Carling (1988).**

3.1 The WEBSOM method

The WEBSOM method is a SOM-based retrieval technique that can be used as a tool especially in exploring document collections but also in various searching tasks (Honkela 1997; Kohonen 1998, Lagus 1996a, 1998, 2000b). This is possible by forming similarity graph of text documents by the SOM principle (Kohonen 2001), i.e., using statistical models (vector space for instance) that describe collections of words in the documents. In this method, a textual document collection is organised onto a graphical map display that provides an overview of the collection and facilitates interactive browsing. With this brief definition of WEBSOM the rest of this chapter is structured as follows. If WEBSOM is the application, what then is SOM? What does the WEBSOM architecture look like? This chapter describes the SOM within the framework of the practical applications, i.e., the case study of improving

retrieval of the document collection belonging to this thesis (the actual applications will be discussed in Chapter 4). It provides a detailed discussion on the SOM algorithm and a step-by-step process of creating a self-organised map of document collection from preprocessing the text (feature extraction and document encoding) to developing a visual browsing interface (Figure 3.1). Other sections attempt to address related issues of how to choose a good document, interpretation, evaluation and use of feature maps in text retrieval. Adding new documents to an existing map is also briefly highlighted.

3.2 SOM algorithm

The SOM (self-organising map) (Kohonen 2001), introduced by Teuvo Kohonen in 1982, is one of the most popular unsupervised-learning[11] neural network[12] algorithm (Lin 1997; Honkela 1997; Kohonen 1998; Lagus 1996, 1998, 2000). Also known as the Kohonen's feature map, the SOM algorithm is quite a unique kind of neural network tool for visualisation of high-dimensional data. In its basic form, the SOM produces a similarity graph of input data (Kohonen 2001). It consists of a finite set of models that approximate an open set of input data and the models are associated with nodes (neurones) that are arranged as a regular grid (Kohonen 1998). The models are produced by a learning process that automatically orders them on the grid along with their mutual similarity.

The map units, neurones, usually form a two-dimensional regular lattice where the location of a map unit carries semantic information. The SOM can thus serve as a clustering tool of high-dimensional data (Han and Kamber 2001). Another important feature of the SOM is its capability to generalise (Kohonen 2001). In other words, it can interpolate between previously encountered inputs.

[11] The training is entirely data-driven and no target results for the input are provided.
[12] A hardware or a computer program which strives to simulate the information processing capabilities of its biological exemplar.

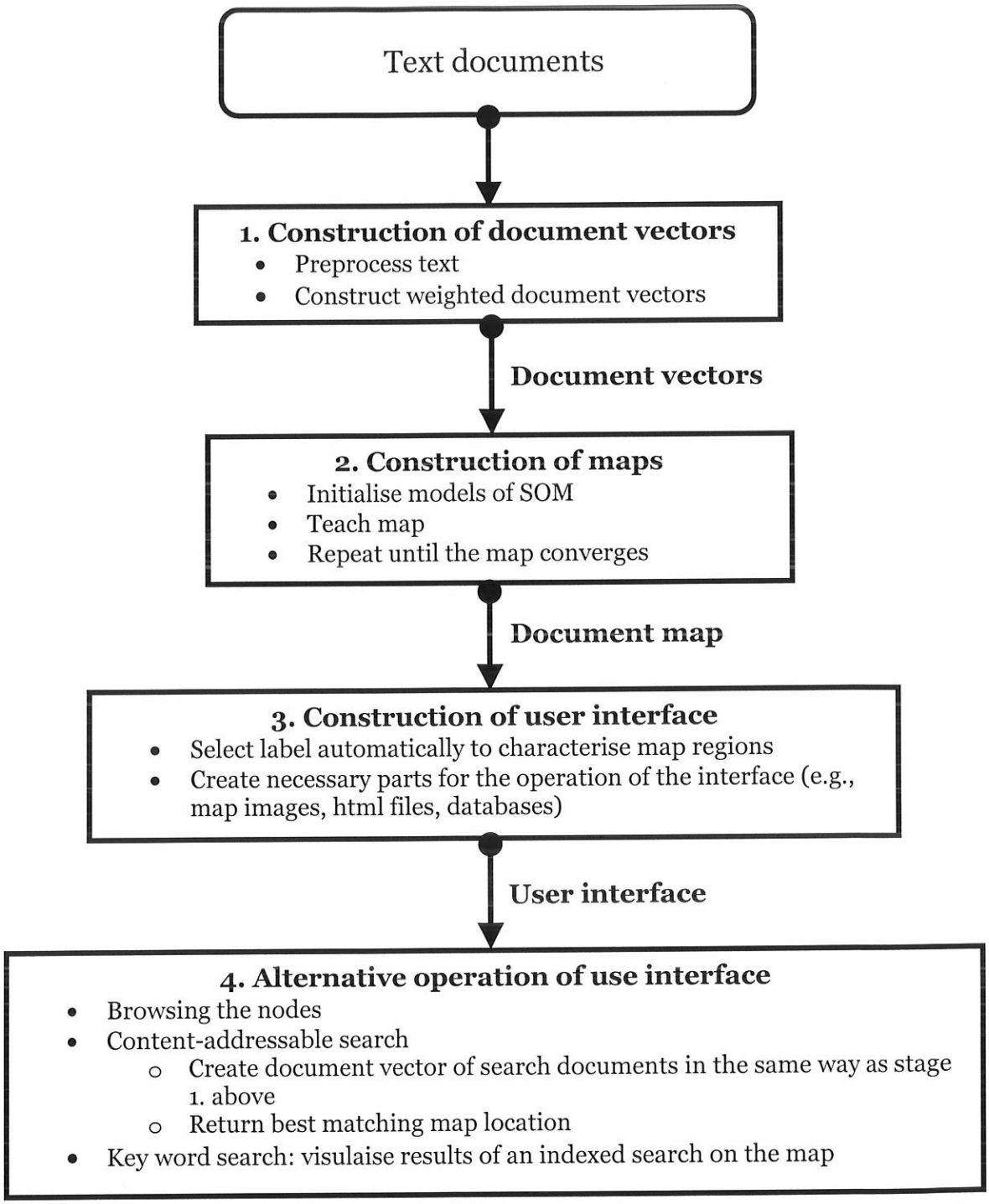


Figure 3.1. *The basic architecture of the WEBSOM method (Honkela et al. 1996; Kohonen 2000).*

The basic SOM can also be visualised like an array, the cells (or nodes) of which become specifically tuned to various input signal patterns or classes of patterns in an orderly fashion (Honkela 1997). Typically interconnected, the number of neurones may vary from a few dozen up to several thousand and are organised on a regular low-dimensional gird. Each neurone is represented by a d-dimensional weight vector $\mathrm{m} = [m_1,...,m_d]$, where d is equal to the dimension of the input vectors. The neurones are connected to adjacent neurones by a

neighbourhood relation, which dictates the topology or structure, of the map. The organised map avails itself readily to visualisation and the properties of the data set can be illustrated in a meaningful manner.

3.2.1 Preprocessing

The SOM requires application of sophisticated feature extraction procedure, for which no evident automatically semantic feature exists (Kaski 1997). A key step in the analysis (Lin 1997; Lagus 1998), feature extraction involves the choice of suitable representation for the data items. The process culminates with representation of document features to vectors before the SOM algorithm can be applied. A general procedure for such a process, based on the vector space model (Salton and McGrill 1983) is used. These procedures involve:

- Identifying a list of terms from a document collection (this may be limited to the document titles, the titles and abstracts, or include the full text of the document)
- Compare the list to a stop word list to delete common terms such as 'and', 'of', 'or' and 'the'
- Use a word-stem procedure to reduce the list to a stem form and remove duplicates
- Remove some of the most and least frequently occurring terms from the list
- Encode the collection based on the remaining list of terms. A vector is created for each document where each component of the vector can be:
  i. A binary digit, i.e. each component will be either 1 or 0 depending on whether the corresponding term appeared in the document or not
  ii. A weight based on the term frequency (the number of times a term appears in a document)
  iii. A weight based on both the term frequency and the inverse document frequency (the inverse of the number of documents in which a term occurs).

The different types of encoding scheme represent a document space differently (Honkela et al. 1996; Kaski 1997; Lin 1997; Lagus 1998). The map displays, based on different types of encoding, may also show different characteristics of the document space.

3.2.2 Computation of Kohonen's feature maps

The basic idea of Kohonen's feature map algorithm takes a set of input objects, each represented by a *d*-dimensional vector, and maps them onto nodes of a two-dimensional grid. Simply stated the mapping procedure (Lin 1997; Honkela 1997; Kohonen 2000) is a recursive learning process that:

- Selects an input vector randomly from the set of all input vectors
- Finds the node (which is also represented by an *d*-dimensional vector called weights) closest to the input vector in the *d*-dimensional space
- Adjusts weights of the node (called the winning node), so that it will more likely be selected again if this input is presented later
- Adjusts the weights of those nodes within a neighbourhood of the winning node, so that nodes within this neighbourhood will have similar weight patterns.

The following example (adapted form Honkela 1997) help illustrate the mapping procedure described above. Assume some sample data set of input variables $\{\varepsilon_j\}$ (Table 3.1) have to be processed by the SOM algorithm.

Table 3.1. *Three-dimensional input data in which each sample vector x consists of the RGB (red-green-blue) values of the colour shown in the rightmost column (Honkela 1997).*

250	235	215	Antique white
165	042	042	Brown
222	184	135	Burly wood
210	105	30	Chocolate
255	127	80	Coral
184	134	11	Dark goldenrod
255	140		Dark orange
...	...	...	...

The set of input samples is definable as a real vector $x = [\varepsilon_1, \varepsilon_2, ..., \varepsilon_d]^T \in \Re^n$ where ε_j is the index sample. Each node i in the map contains a model vector $m_i = [m_{i1}, m_{i2}, ..., m_{id}]^T \in \Re^n$, which has the same number of elements as the input vector x. Assuming a general distance measure between x and m_i The SOM algorithm then takes the input vector x and compares it with all the model vectors m_i and identifies the best-matching unit (BMU) (the

winner node) on the map, i.e., the node where the model vector is most similar to the input vector. The model vectors of the winner and its neighbouring nodes are updated on the map towards the input vector (Figure 3.2).

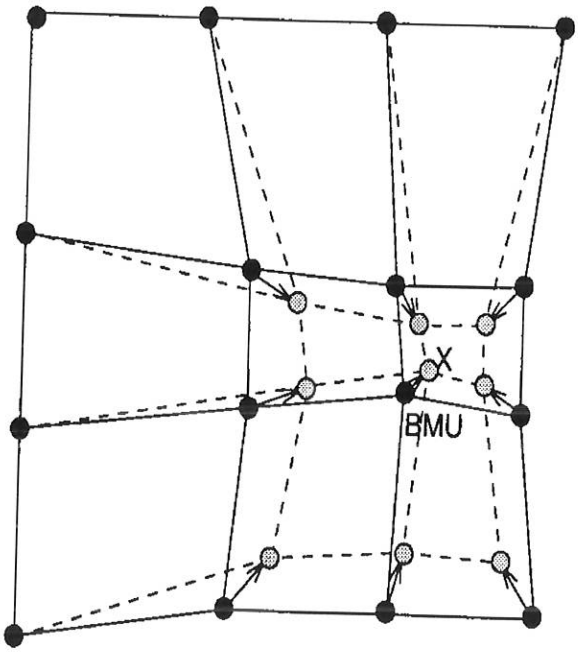


Figure 3.2. *Updating the SOM and its neighbours towards the input sample marked with x. The solid and broken lines correspond to situation before and after updating (Vesanto et al. 2000).*

This process goes through many iterations until the mapping is ordered and descriptive of the distribution of x or the adjustments all approach zero (Kohonen 2000). (Note that the model m_i need not be vectorial variables. It suffices if the distance measure is defined overall occurring x items and sufficiently large set of models m_i. SOM imposes two control mechanisms to ensure map convergence (Lin 1997). The first is the updating parameter. It approaches to zero as the number of iterations increases.

The second is the neighbourhood structure that shrinks gradually during the process. A large neighbourhood will achieve ordering and a small neighbourhood will help to achieve a stable convergence of the map (Kaski 1997; Kohonen 1998). By beginning with a large neighbourhood and then gradually reducing it to a very small neighbourhood, the feature map achieves both ordering and convergence properties (Lin 1997). Lin (1997) notes, in practice, the process is usually divided into two phases. The first takes relatively fewer iterations and starts with a large neighbourhood. The second phase is for fine-tuning of the

feature map. It starts with both small neighbourhood and small updating parameters, and requires many more iterations to complete.

In other words, these whole process defines a mapping from a d-dimensional input data space onto a regular two-dimensional array of nodes. The input i of the map is associated with the d-dimensional reference vector $m_i = [m_{i1}, ..., m_{id}]^T$, where d denotes the dimension of the input vectors. The reference vectors together form a codebook that are connected to adjacent nodes by a neighbourhood relation which dictates the topology or the structure, of the map. The most common topology (Figure 3.3) in use is rectangular and hexagonal (Lagus, Kohonen, others). Adjacent nodes then belong to the neighbourhood N_i of the neurone i. In the basic SOM algorithm, the topology and the number of nodes remain fixed from the beginning. The number of model vectors determines the granularity of the mapping, which has an effect on the accuracy and generalisation of the SOM (Kohonen 2000; NNI 2001).

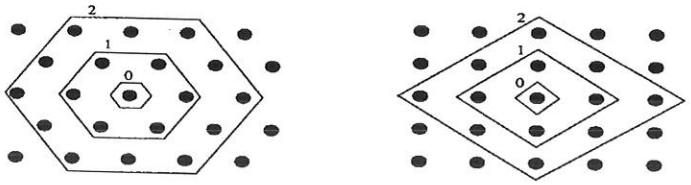


Figure 3.3. *Neighbourhoods (0, 1 and 2) of the centre most unit: hexagonal gird on the left, rectangular on the right. The innermost polygon corresponds to 0-, next to the 1- and the outermost to the 2-neighbourhood (Vesanto et al. 2000).*

During training, the SOM forms and controls an elastic net that folds to approximate the density of the data. The reference vectors in the codebook drifts to the area of the map with high density of input data. Eventually, only few codebook vectors lie in the area where the input data is sparse.

The SOM algorithm learns (Kohonen 2000; NNI 2001) in the following way:

1. One sample vector x is randomly drawn from the input data set and its similarity (distance) to the codebook vectors is computed by using e.g. some distance measure

$$|x - m_c| = \min_i \{|x - m_i|\}$$

2. After the best matching unit (BMU) has been found, the codebook vectors are updated. The BMU itself as well as its topological neighbours are moved closer to the input vector in the input space i.e. the input vector attract them (see Figure 3.2 above). The magnitude of the attraction is governed by the learning rate ($\propto$). As the learning proceeds and new input vectors are given to the map, the learning rate gradually decreases to zero according to the specified learning rate function type. Along with learning rate, the neighbourhood radius decreases as well. The update rule for the reference vector of unit i is:

$$m_i(t+1) = \begin{cases} m_i(t) + \alpha(t)[x(t) - m_i(t)], & i \in N_c(t) \\ m_i(t), & i \notin N_c(t) \end{cases}$$

3. The steps 1 and 2 together constitute a single training step and they are repeated until the training ends. The number of training steps is fixed prior to training the SOM because the rate of convergence in the neighbourhood function and the learning rate is calculated accordingly.

After the training, the map is topologically ordered, i.e., n topologically close input data vectors are mapped to n adjacent and/or even to a single map node.

As a result of the recursive learning process, a grid of nodes with randomly assigned weights will be transferred into an orderly feature map that reflects the patterns or structures of the input. The mapping process not only categorises the input data but also preserves local neighbourhood relationships among the input data. It creates a 'continuous elastic surface' and distributes the input data on the surface based on their statistical patterns (Kohonen 2001).

The algorithm has been applied to various practical problems in the areas of speech recognition, robot arms control, optimisation problems and analysis of semantic information (Kohonen 2001). Early applications of the algorithm mostly demonstrated that the feature map preserves metric relationships and the structure of input patters (Lin 1997; Kohonen 1998). The application of the feature map to cognitive information processing (Kohonen 2001) particularly offers the possibility 'to create in an unsupervised process topographical representations of semantic, non-metric relationships implicit in linguistic data.' It has also

been possible to form similarity graphs of text documents by a document organisation, searching and browsing system conveniently call the WEBSOM.

The original WEBSOM (Kohonen 1996) was a two-level SOM architecture. The first experiment (Honkela et al. 1996) was done on close to 5000 neural network articles from a Usenet newsgroup that appeared during the later half of 1995 and up to June 1996. The WEBSOM method has since been applied to the over 6.8 million patent abstracts (the largest to date) (Lagus 1998) that were available in electronic form. On a much smaller scale, WEBSOM has also been applied to classify Finish news articles.

3.3 User interface for document maps

The practical purpose of developing the WEBSOM document map is to provide an interface to a collection so that any one could conveniently explore. The visualisation is meant to help in forming a general view of the domain, as well as to guide exploration towards potential interesting particular areas (Honkela et al. 1996; Lagus 2000b). In addition, a convenient and efficient strategy of moving from general view to specific details and back have been designed, including methods of interaction and a strategy for providing the user with a sense of location and context (Lagus 2000b). Suitable visual and textual means for conveying information about the content of the map is as well developed (Kohonen 2000).

3.3.1 Navigation interface

The WEBSOM navigation interface consists of an image of the whole of document map, a hierarchy of zoomed pieces of the map at various zooming levels, and a set of HTML pages, image map files and CGI or ASP (Active Server Page) scripts. Zooming in is achieved by pointing and clicking with the mouse at the desired location on the map image. Horizontal movement to nearby areas as well as panning out is carried out by clicking on a compass image. On the more detailed zoom levels white dots mark units of the regular, map gird. By clicking near a dot the list of documents associated with the map unit is accessed. From the list, individual documents can be selected for reading by clicking on the title.

3.3.2 Visualised document map

The WEBSOM document maps are visualised using two methods, a smoothed version of the unified distance matrix or U-matrix and a smoothed document density diagram (Lagus 2000b). In the U-matrix visualisation dark colour corresponds to a considerable difference between the model vectors of neighbouring map units, where as a brighter colour signals similarity between neighbours. In contrast, in the density diagram light colour denotes a large number of similar documents and dark colour an emptier area. Due to the relationship in SOM between density of model vectors and density of documents, both methods visualise the cluster structure to some degree, although U-matrix more faithfully (Lagus 2000b).

The smoothed document map landscape carries out the following functions: 1) it describes the document density at each area and 2) it provides texture that can help maintaining a sense of location and context across movements and across dynamic visualisations (Lagus 2000b).

3.3.3 Labelling the map display

Interpretation of a document map display can be aided by labelling the display with a selection of descriptive words that characterise regions of the map. The labels can be utilised for multiple functions: 1) to describe the underlying area, contrasting it against the rest of the map, 2) collectively to summarise aspects of the collection, 3) and in navigation to act as landmarks or anchor points that help orientation by providing reference points during transitions across views that have different resolutions.

An automatic method has been introduced (Kohonen 1999) for selecting descriptive terms suitable for characterising textual clusters, individual map units and document map regions. The method was validated against human-assigned descriptors (Lagus 2000b).

This method has been formalised by a measure that compares the word's frequency to other word frequencies in the cluster, and also to its own relative frequency generally in the collection. The measure of goodness $G(w, j)$ for a word w to characterise cluster j is defined as:

$$G(w, j) = \frac{F_j(w) F_j(w)}{\sum_i F_i(w)}$$

where *Fj (w)* is the proportion of the word w in cluster j.

3.3.4 Search facility

Especially when browsing large document maps it may be difficult to decide where to start browsing the map. A search facility can be implemented to provide suitable starting points for exploration (Kohonen 2000). The description of interest written by the user—either a whole document or a few words—is encoded (see section on document encoding above) as a document vector and a number of best-matching map units are marked on the display with circles the radius of which convey the goodness of the match. It should be noted that this facility does not perform document retrieval, i.e. return the best-matching documents, but only returns the best-matching map units (Lagus 2000b).

The facility is implemented using a client-server architecture: A search server hold the map reference vectors in memory and when a search is initiated, a client program encodes the query as a document vector, passes it to the server and requests for a number of the best-matching units. Upon receiving the results the client draws them on existing static map images, constructs an appropriate HTML page and returns it to the WWW browser.

3.4 Choosing a good map

The stochastic-based SOM learning process (see Section 3.1.2) allows some variations in the learning results (Kaski 1997). Thus, to ensure good quality map several maps may need to be computed and the best map chosen according to some cost function specific to the size and topology of the map. Note that this limits the possibilities to use a cost function to compare maps of different sizes or neighbourhood function (Kaski 1997; Lagus 2000b). The value of the cost function is inversely proportional to the size of the map and increases as the width of the neighbouring function increases.

3.5 Interpretation, evaluation and use of the maps

Interpretation: Some general methodology may aid in the interpretation of maps (Kaski 1997; Lagus 2000b), although the interpretation of the map should be predominately local (because SOM attempts above all to preserve local structures) based on the local relations of the data items on the map. The global structure is also useful. Different properties of the reference vectors and the data items can be visualised on the map display to help in the interpretation.

Another method that helps in the interpretation, provided that some external information like class labels are available, is to plot the labels on the organised map. Lagus (2000b) utilised a measure based on an external topical classification of documents, best described as the purity of map nodes, defined as the proportion of documents that fall into a map unit where their own class form a majority for evaluating document maps. The classification accuracy then indicates how well the classes are separated on the map, and the classification accuracy of new samples measures the generalisability of the results. If the distribution of the known classes are overlapping such displays can even be used to explore the degree of overlap in different types of samples. It may then be possible to gain insight to whether the classes actually co-exist or whether new kinds of features should be added to the data items to make the classes easily separable.

Evaluation: Although good evaluation methods for measuring the quality of visualisation, exploration and navigation are very difficult to define (Lagus 2000b), the quality of the map display may generally be evaluated by an expert in the application area (Kaski 1997). User studies may also be required until a more direct, automatically applicable measures are determined (Lagus 2000b). If samples having known classes are available, it is potentially useful to try to classify the samples using the map. Each map unit is labelled according to a majority voting of the samples, after which all samples that are projected into a unit (node) are classified according to its label.

The generalisability of the results could also give some indication of the quality of the mapping. Generalisability could be measured as the sensitivity of the map to small variations in the input data, caused for instance by adding artificial noise.

Use of the organised map: The illustrations formed by the SOM can be used as tools for gaining insight into a data set. They can also be used to summarise data sets, together with explorative research, or even as a decision support system (Kaski 1997).

The SOM can be used in facilitating exploration of a data set, searching for known kinds of data, filtering of new incoming data, as well as visualisation of the results (Lin 1997). The SOM can also be used for extracting clusters automatically and for rule extraction (Kaski 1997).

3.6 Adding new documents

New documents can be inserted onto an existing map simply by locating the best-matching map unit for each document (Lagus 2000b). However, in a non-stationary document collection where new topic areas and terms are introduced, after a while the map may not be such a good representation of the collection. An intuitive reason for this is that in a very high-dimensional and very sparse space an unseen document will not be near any area of the two-dimensional map unless its own topical domain is discussed by documents that contributed to the map construction. In a non-stationary collection the map should therefore either be incrementally adapted or fully re-calculated after a time (Lagus 2000b).

3.7 SOM implementation tools

A number of SOM implementation tools that concentrate on the standard SOM have over the years been developed. The first public domain program published by groups that have experience of some practical SOM applications was SOM_PAK (http://www.cis.hut.fi) (Kohonen 2001). Intended for a general-purpose development tool, SOM_PAK was released

for the first time in 1990 by the Laboratory of Computer Science of Helsinki University of Technology (LCIS/HUT).

Later in 1996, LCIS/HUT also released the first version of SOM Toolbox (http://www.cis.hut.fi/projects/somtoolbox), designed for the MatLab (http://www.mathworks.com) system to handle most practical applications of the algorithm and to provide for a better platform for experimentation, as well as versatile visualisation. Other SOM implementation tools also include Nenet (Neural Networks Tools), the software used int his project, and Viscovery SOMine. The former (also commericially available) was first released by graduate students, the Nenet Team of HUT in 1997. It was intended to be user-friendlier than the artificial neural network programs of that time (Kohonen 2001). While SOM_PAK has been designed for very large and computationally heavy professional tasks, Nenet and SOM Toolbox are best suited for relatively small-scale problems, which illustrate the use of SOM and as a compromise between size and nature of task, respectively.

Viscovery SOMine, on the other hand, is a commercial SOM software package produced by Eudaptic GmbH in Austria. It is considered as user-friendly, flexible and powerful, especially in statistical problems: financial, economics and marketing applications. SOMinePro is the only SOM implementation tool that provides *some* features for the processing of text files.

Although it is very difficult to review the plethora of software packages that include the SOM along with many other algorithms, it can generally be said that these packages do not normally contain as many SOM features (Kohonen 2001). These packages are also seriously criticised for questionable training sequences and lack of monitoring programs to test the quality of maps (Kohonen 2001). Nenet was therefore chosen for the task on the ground that it is a standard SOM implementation and is sufficient for the development of the prototype. The professional version of Nenet used in this study also does not put limit to the dimension of the data vector.

Chapter Four: The document map of ILRI publications

Contents

4.1 The document map of ILRI publications

The current availability of large collections of full-text documents in electronic form emphasises the need for intelligent information retrieval techniques. Especially in the rapidly growing digital libraries and distributed access, it is important to have automatic methods for exploring document collections. In this study, the WEBSOM method is used with a quarter of century of research publications maintained by the International Livestock Research Institute for this task. The Self-Organising Map (SOM), also known as Kohonen's feature map (a means for automatically arranging high-dimensional statistical data), is used to position encoded documents onto a map that provides a general view into the text collection. The general view visualises similarity relations between the documents on a two-dimensional map display, which can be utilised in exploring the material rather than having to rely on

traditional search expressions. Similar documents become mapped close to each other providing an intuitive mechanism and ease of access for maximising the institute's digital information and knowledge resources particularly for users with limited domain knowledge.

Within the framework of the basic architecture of the WEBSOM method (discussed in the previous chapter), chapter four essentially presents the results of the experiment in this study together with a discussion on the prototype developed as part of the effort to help categorise the ILRI research publications and improve interactive exploration. The highlights of the entire process involved:

- Writing the VB code that is linked to the publications database.
- Extract the document features from the tittles and abstracts and form a data vector from each document in the training dataset.
- Label each vector with the document ids of the abstract so that it could be possible to know from which abstract and document the vector has been formed.
- Write the vectors with the labels to a plain ASCII data file format understood by Nenet. Note that Nenet only recognises a plain ASCII data format.
- Use Nenet to train a map with the vectors. Note that the training and preprocessing parameters one uses with the data is essential to get good maps.
- Use the Nenet auto-labelling feature in the test phase to label the neurons. The distribution of the labels provides a rough idea of the goodness—evenly distributed over the map area—of the map.
- Create some test vectors that 'resemble' the original vectors from the rest of the document titles and abstracts.
- Test and see how well the map classifies the vectors.
- Write appropriate ASP codes and design the necessary HTML pages of the browsing interface.
- Test relative merits of the special presentation of the WEBSOM to the traditional keyword based search.

Sample maps have as appropriate been included to provide for the completeness of the discussion as well as illustrate the concepts. Subsequent sections in this chapter provide a detailed treatment of each of the above procedures. A brief description of the structure and organisation of the current full-text InMagic database of the institute's research publication sets the scene by providing an initial idea of the nature of the document collection. Other section discusses the complex document feature extraction procedure used in this study, a discussion of the rigorous process of training and test data preparation and development of the web-based user interface and document databases in that order. Highlights of the results of the experiment to improving interactive exploration and classification power of the SOM are also addressed later in the chapter. Referred annex for details of the logic behind the various functions and procedures used. An alternative testing technique to complement the findings in this research were users feedbacks. A brief discussion on the initial reactions of a few users of the web-based user interface is provided towards the end of the chapter.

4.2 The ILRI publications database

The current ILRI publications database contains close to 4000 documents classified into several subject codes based on FAO-AGRIS/CARIS (International Information System for the Agricultural Sciences and Technology/Current Agricultural Research Information System) classification scheme. This full-text database contains publications in two major languages: English and French. The text used in this study is English which accounts for well over 95 percent of the collection.

These publications range from the very technical research publications of DNA-based breed modelling and characterisation of animal genetic resources to production systems and nutrition and feed resources studies. Other general subject codes used in the classification scheme include animal science, production and protection (animal husbandry, animal feeding, animal health and disease, etc.) agricultural economics and policies, plant science and production (crop husbandry, cropping patterns and systems), natural resources and the

environment and processing of agricultural products. These research publications are assigned to one or multiple subject codes by a team of human indexers. These publications are further categorised by 96 subject and descriptive index fields. The later classification among others provides document with a unique identification number, publications category, project codes, authors information, titles, sub-titles, corporate source, abstract, publication date, descriptors, etc. Only seven of these fields (document ID, title, subject code, abstracts, author, descriptors and year of publication) were considered for this exercise and were extracted from the database for further processing and conversion to MS Access.

4.3 Preprocessing

Before text documents are presented as vectors to act as input to SOM, a sophisticated preprocessing (feature extraction) procedure (Appendix I) has been applied to automatically extract the lexical profiles of terms (the rundown of terms within a document and in the collection) to represent the data items. The feature extraction procedure takes the form of filters to remove words low in content (stop words) from the text and provide vectors representing the key terms in the titles and abstracts suitable to represent semantic features of documents. The general procedure used in this process involved automatically:

- Identifying a list of terms from the document titles and abstracts
- Comparing the list to stop words (separately developed from standard Verity Stop List and adds the words from the DTIC-DROLS stop word list) to delete common terms such as 'and', 'of', 'or' and 'the'
- Using a Visual Basic (VB) implementation of Porter word-stem procedure to reduce the list to a stem form and remove duplicates (the original BASIC program of the Porter word-stemmer algorithm implementation, now integrated to VB, is a free code downloaded from the Internet)
- Removing some of the most and least frequently occurring terms from the list and encoding the collection based on the remaining list of terms.

The preprocessing phase started off with the entire full-text document database of 3940 records. The number of records considered was later reduced to 987 records, as the rest of the documents in the database have no abstracts (See create abstract data refined function[13], Appendix I). The whole vocabulary contained over 57 thousand entries/terms. The size of the vocabulary was later reduced to 20 thousand when all the words were converted to their base form by a VB implementation of the Porter English word stemming algorithm (see word stemmer function, Appendix I) and the words occurring less than 10 and over 50 times in the whole corpus, as well as a set of common terms in a stop word list of 437 words were removed. Note that the lower and upper threshold values were determine based on the results of an evaluation of the significance of the terms as index terms by a team of indexers at the institute. The most common terms that were not powerful enough to discriminate the collection and those terms that are very specific to only a few of the collection were removed.

The next phase in preprocessing is to create datasets for both the training and testing the neural network algorithm (SOM). The document database was thus classified into two sets of records: the training and test datasets (see prepare data set function, Appendix I) based on the classification scheme adopted by the institute and rigidly applied to the whole collection. Note that the training and preprocessing parameters one uses with the data is essential to get good maps and test vectors need to 'resemble' the original vectors. The training data set contained 347 records (approximately one third of the entire the 987 documents considered for this research) involving all the subject codes and constitute a little over half of the entire single subject code publications. To ensure that the sample is representative of the collection, documents from all the different subject codes used to classify the entire 987 publications are included in the training dataset. The automatic classification procedure also includes publications when they are the only once in a particular subject code. The balance of 640 records, consisting of publications assigned to multiple subject codes and the remaining half of single subject coded publications were put aside for testing.

[13] Note that function names are provided in the text as phrases for sense while they appear as one word in the annexed codes in the report.

This classification of dataset was followed by the rigorous word generating function (see generate word function, Appendix I). This automatic procedure was separately applied to both the training and test datasets. The function works on the individual tables for the two sets with the document ID, title, abstract and subject code fields. The procedure extracts the features from both the titles and the abstracts, removes symbols, single character word, checks if the term is not a stop word (Table 4.1) by using the various English word delimiters (like space, comma, semicolon, etc.). This stage also removed numerical expressions.

Table4.1. *Partial of the stop word list.*

stopword
a
Ababa
about
above
according
.
.
yet
you
your
yours
yourself
yourselves

Note that the stop word list was separately compiled by the author from the standard Verity Stop List and adds the words from the DTIC-DROLS stop word list and the Porter stemming algorithm.

The word generating function further computes the within document term frequency (tf) and collection frequencies of the individual terms and calls a Porter stemming algorithm to convert words into their base form. It is an iterative process that for each term in a document computes the tf and idf organises the records into a table (Table 4.2) with the document id (docid), word, frequency in document (freqindoc) and collection frequency (CF) fields.

To avoid repetitive computation and unnecessary checking of term weights of similar words in different documents, the collection frequency field is assigned o values until later processes are completed and the distinct terms have been identified. The function iteratively works by maintaining the lexical profiles of a term within a document in a temporary table

with word and frequency fields (Temporary table in the prototype database) until it permanently posts it to the vector table (Table 4.2) for all the documents in the dataset.

Table 4.2. Partial view of the vector table.

docid	word	frequencyindoc	CF
2	ADOP	1	0
2	AFRICA	1	0
2	AGRICULTUR	2	0
2	AGROINDUSTRI	1	0
2	CROP	1	0
2	DAIRY	1	0
2	DISCUSS	1	0
.	.	.	.
.	.	.	.
1867	SUBJECT	1	0
1867	TREATMENT	12	0
1867	TREND	1	0
1867	UNDERSTOOD	1	0
1867	VEGET	4	0
1867	YEAR	4	0

The training dataset table keeps the profile of 17,687 terms as they appear in the various documents in the collection. Note that the same procedure is applied to the test dataset. The only difference being the actual data input parameters for the function. While the function works with the training dataset table when preprocessing data for training the map, it uses the test dataset table to generate the lexical profile of documents that represent the test dataset.

Once the terms have been identified and their within document frequencies computed, the next step is to identify the distinct words to help in computing the collection frequency. The table above would be further processed by yet another function (see distinct all from vector table function, Appendix I) that selects all the distinct words from the collection and maintains them in a single table with 0 collection frequency field (Table 4.3). The number of distinct words table is then created with 4056 different words.

Now the actual collection frequency of the individual terms can be generated (Table 4.4). The collection frequency of a term is then computed (see compute CF function, Appendix I) by keeping track of the distinct terms from the list of distinct terms and progressively counting the number of documents the term occurred. This information would be

temporarily maintained with the distinct terms until it is later updated to the vector table. The collection frequency of these terms ranges from 1 to 128.

Table 4.3. Partial view of the list of distinct terms.

word	CF
AA	0
ABANDON	0
ABATTOIR	0
ABIL	0
ABILITY	0
.	.
.	.
ZIMBABWE	0
ZINC	0
ZIPPER	0
ZONE	0
ZONAL	0
ZOOTECHNIQU	0
ZORNIA	0
ZYGOT	0

Table 4.4. Partial view of the list of distinct terms with their collection frequencies.

word	CF
AA	1
ABANDON	1
ABATTOIR	4
ABIL	1
ABILITY	5
.	.
.	.
ZIMBABW	9
ZINC	2
ZIPPER	1
ZON	22
ZONAL	1
ZOOTECHNIQU	5
ZORNIA	1
ZYGOT	1

In the interest of maintaining related information in one table, the collection frequency (CF) field in the vector table is then updated (Table 4.5). A simple SQL update query (see distinct all from vector table function, Appendix I later integrated to the VB code as a function) updates all the zeros in the vector table (Table 4.2 above) with the actual collection frequency values of the individual terms.

Table 4.5. Partial view of an updated collection frequency field vector table.

docid	word	frequencyindoc	CF
2	ADOP	1	4
2	AFRICA	1	58
2	AGRICULTUR	2	22
2	AGROINDUSTRI	1	1
.	.	.	.
.	.	.	.
1867	SUBJECT	1	5
1867	TREATMENT	12	30
1867	TREND	1	9
1867	UNDERSTOOD	1	1
1867	VEGET	4	6
1867	YEAR	4	32

Further inspection of the table above (Table 4.5) and the collection frequency of the individual terms show the need to determine threshold values to eliminate some of the most and least frequent terms in the collection. A high value for *idf* coefficient (collection frequency) is indicative of a word common in the corpus under examination. By disregarding words with *idf* coefficient lower than 10, many closed class and other terms common in general language are automatically removed from the set. Another set of most frequently occurring words were also removed from the table by setting a threshold *idf* coefficient value of greater than 50. The second ground of terms are hardly used by the institute's indexers as descriptors. Some of these terms include Africa, feed, study, discuss, cattle, result, level, sheep, etc. This procedure further creates a new table with an additional term-weight field to prepare the data to yet another important preprocessing phase. The modified vector table (Table 4.6) essentially maintains the structure of the updated vector table (Table 4.5 above). The number of records for further processing has now been reduced to 8257 records sorted by the document id and word and the additional term-weight field.

Table 4.6. Partial view of the modified vector table with the most and least frequent terms removed.

docid	word	frequencyindoc	CF	Weight
2	AGRICULTUR	2	22	113
2	CROP	1	37	216
2	DAIRY	1	24	140
2	FEED	6	49	199
1867	RESPONS	2	34	175
1867	SEMI	1	12	70
1867	SIMILAR	1	35	205
1867	SOIL	8	22	83

Table 4.6. Partial view of the modified vector table with the most and least frequent terms removed.

docid	word	frequencyindoc	CF	Weight
1867	STOCK	4	17	76
1867	TREATMENT	12	30	101
1867	YEAR	4	32	143

Term-weights were then computed for these records based on two measures: the within document frequency (computed with the generate word function) and collection frequency (computed by the function compute collection frequency). See compute weight for modified vector table function, Appendix I, for details. The procedure then updates the modified vector table with the actual term weights following a user input of the number of documents processed. A vector then is created for each document where each component of the vector is the weight—the measure of the distribution of the word in the whole collection—based on both the term frequency and the inverse document frequency (the inverse of the number of documents in which a term occurs). Note that the distinct terms have yet to be extracted from the modified vector table to help determine the dimension of the final document vector and also determine the different vectors and describe the document collection only by these distinct terms. A similar procedure, applied above to select the distinct words form the vector table and in computing the collection frequency, is again used to extract the distinct terms from the modified vector table or the remaining records after the most frequent and least frequent terms have been removed.

The distinct terms table (Table 4.7) with 421 words ordered alphabetically now serves as the landmark to encode the entire research publications (both the training and test dataset) and is used in creating the term-document matrix to serve as input to the SOM. The two sets of documents were then encoded based on these distinct terms.

The final phase of the preprocessing involves creation of the actual data vectors in a text file (format compatible with Nenet) document-term matrix (Figure 4.1, see also the create vector in matrix form function, Appendix I). The training and test datasets were written to a file with the following names: 'trainingdata.dat', 'testdataset.dat', 'Fo1testdata.dat' and 'Lootestdata.dat'.

Table 4.7. Partial view of the final list of distinct terms used to describe the collection.
word
ACCESS
ACID
ACTIVE
ACTIVITY
.
.
WET
YEAR
YIELD
ZEBU
ZONE

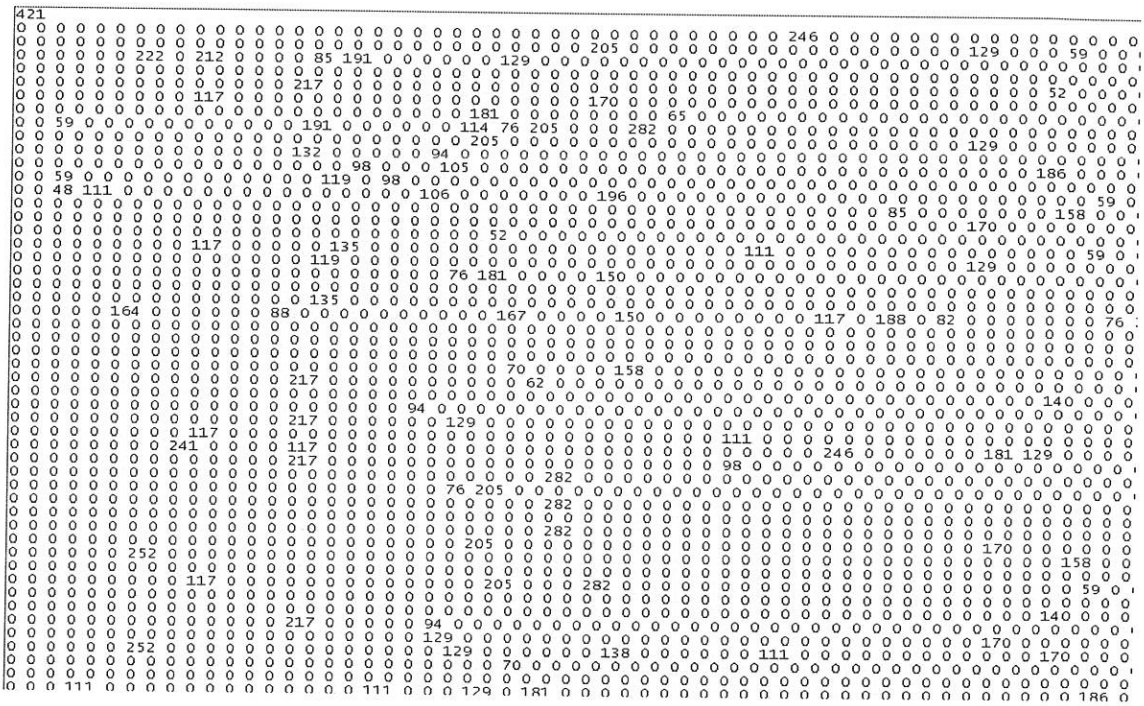


Figure 4.1. *An example of a 421-dimension data vector used in this study. The first line represents the dimension of the data vector. Each raw represent documents and the column distinct words used to encode the set. Words or document ids may be appended to each line to help identify which vectors represent which document and could be used for auto labelling map regions.*

This function again was implemented by a procedure that for each document and each word in the document checks where the word is on the list of distinct words (note that this list is sorted by word) and goes to that particular column and writes the term weight for that term for that particular document. Vectors representing the titles and abstracts were created on the basis of a lexical profile of the terms in the text determined above by the two measures: the within document frequency of a term and inverse document frequency. As a result, a vector of 421 dimensions is created for each document, where a component is the

weight of the corresponding terms in the document. If a term, that serves as a column heading in the term-document matrix, does not have an associated weight in a document it gets a zero value. Note also that each row in the document-term matrix represent a document.

A similar automatic and step-by-step procedure is applied to the test dataset to extract the document features and create the document-term matrix. The same function manipulating different datasets generate the word and compute the term and collection frequencies and later the tf*idf term weights. The test dataset only consisted of the remaining documents from two randomly picked subject codes (12 documents assigned to the F01 (Crop husbandry) subject code and 18 documents assigned to L00 (Animal genetics and breeding) subject code) that were not included in the training dataset. With exactly the same structure to the vector table for training, the vector table test keeps track of the lexical profile of the various terms in the test dataset. The vector table for test dataset is input into a similar set of functions that selects and creates the list of distinct terms, computes and updates collection frequencies as well as weights terms all for the test sets. An additional process introduced to reduce the computational load of computing collection frequencies and term weights for the test datasets was to first identify all the terms that are commonly shared by the list of distinct terms used to represent the dimension of the vector for the training set and the list generated for the test. Collection frequencies were thus computed only for the terms that are commonly shared by both sets. A modified vector table for test data was then created (see create modified vector table test function, Appendix I). Similar threshold values were applied to remove both the rarest and most frequent words in the collection before weights were computed. Term weights were then computed for the remaining set of records and a document term matrix was separately created for each set of 12 and 18 documents from the two subject codes.

A similar procedure to that applied for the training dataset has been applied to create the data vectors in a text file. Note that to reduce complexity of the program that implements these rigorous procedures to preprocess the data and extract the document features,

functions commonly shared by both datasets accept parameters (table names in most cases) instead of fixed values. This has helped to use the same functions to process both the training and test datasets. Note also that all the functions are combined into one main function to automate the whole process.

4.4 Formation of the Kohonen feature map

The preprocessing phase is only where the initial beef lies. The following section discusses yet another essential aspect of the study—the computation of the Kohonen feature map. The basic idea of Kohonen's feature map algorithm takes a set of input vectors, each represented by the 421-dimensional vector and maps them onto nodes of a two-dimensional grid. The mapping procedure is a recursive learning process that:

- Selects an input vector randomly from the set of all input vectors
- Finds the node closest to the input vector in the 421-dimensional space
- Adjusts weights of the winning node, so that it will more likely be selected again if this input is presented later
- Adjusts the weights of those nodes within a neighbourhood of the winning node, so that nodes within this neighbourhood will have similar weight patterns.

Computation of the two-dimensional Kohonen feature map by Nenet involves three important phases: initialisation, training and testing.

4.4.1 Initialisation

A new Nenet map is first initialised with the appropriate parameters. The SOM X and Y dimension (specifies the X and Y dimensions of the self-organising feature-mapping algorithm to organise the vectors) were set to 36 and 40, respectively. The dimension provides for an average of three documents per node for the entire collection of the institute (3940). The structure used was hexagonal map topology and a bubble neighbourhood function (the function that describes how the neighbourhood is taken into account in the

SOM algorithm) and a linear initialisation type with random seed value of 0. No preprocessing method was specified for the initialisation.

The map was then initialised by the training dataset with 421-dimensional document vector (Trainingdata.dat[14]) as input and a codebook was generated.

4.4.2 Training

The training procedure follows from the initialisation type selected in the map initialisation phase and requires choice of appropriate training parameters based both on the nature and size of the data vectors. This study used a linear initialisation and the training proceeded in one phase because this initialisation type does the ordering of the reference vectors of the map neurons. (Note that if the random initialisation type were used, the training would have involved two phases.) Learning rate (´alpha´) was set to 0.05. The value of learning rate was decreased after one million training cycles (length of the training measured in steps, each corresponding to one data vector) from 0.05 to 0.02 and remained 0.02 for a further 100 thousand training cycles.

The size of the neighbourhood radius in the beginning of training was set to 10 and it stopped at 1 when the training sequence ends. The map training session then started with a 421-dimension data vectors representing the 347 publications in the training data set. This process goes through many iterations and is left to proceed until the mapping is ordered and is descriptive of the distribution of the data vector or the adjustments all approach zero. Nenet provides a convenient view of both the progress in the mapping and the adjustment to the codebook. The training parameters (learning rates and neighbourhood radius) used in this study were based on the experiences of other experiments on other document collection and intuition. Honkela et al. (1996) for instance used a 24 by 32 dimension map for the full text of 4600 Usenet newsgroup articles. Lagus et al. (1998) organised 10,000 patent abstracts from the European Patent Office's collection onto a document map of 28 by 36 units.

[14] Trainingdata.dat contains 347 data vector samples from the document collection. The data dimension is 421.

In addition, to check the efficiency of the parameters used in this study, map training was initially conducted using different training parameters: learning rates and training regimes. Several training sessions were run using different parameter values and without a specific preprocessing method and/or where ranges were used. In one case, for instance, learning rate (α) was set to 0.02 throughout and in another α was set to 0.08 initially (gave the map high opportunity to move initial clusters) but was then set to 0.02 for the final few training cycles. The ordering of the data vectors differs with each learning rate, training regime and map dimension giving minimal chance of comparison of the various training sessions although the SOM could still efficiently classify the data.

The entire process of training the document map took about 36 hours on a Pentium III-processor Dell OptiPlex G10 workstation. Training the map took 100 percent of the CPU time for the entire period of the training. The amount of main memory required was about 120 MB.

4.4.3 Testing

In order to get an idea of the quality of the ordering of the final map, it was necessary to measure how the different data vectors (representing different documents) were clustered on the map by running different sets of input data that were not included in the training data set. A final phase in the formation of a Kohonen feature map, testing the efficiency of the ordering of similar pattern data vectors is required to make sure that the map has successfully adapted to the input data. When the winning map nodes were automatically labelled by the document ids, the resulting accuracy (the purity of the nodes) was resounding. Three separate tests runs were conducted using data vector of documents from two randomly selected subject codes (Crop husbandry (F01) and Animal genetics and breeding (L10)) and data vectors representing the rest of the 640 publications that were not included in the training dataset before the entire data vectors for the document collection were mapped.

The testing data files (testdataset.dat, F01testdata.dat and L10testdata.dat) used contain the remaining 640, 18 and 12 data vector of documents representing the balance of the single

and multiple publications, L10 and F01 subject codes, respectively. Note that the two smaller sets are also included in the larger test dataset. The trained map was first tested with documents from the F01 subject code (Figure 4.2).

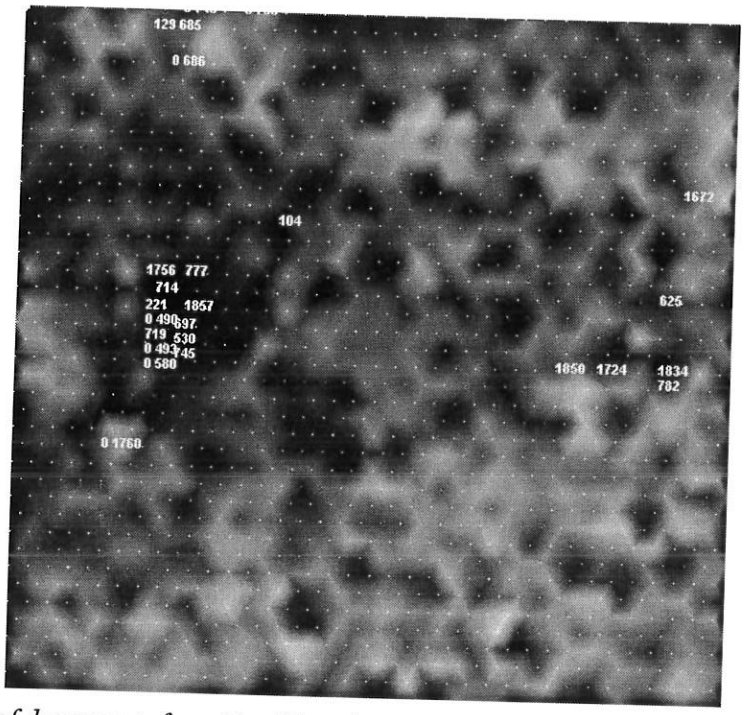


Figure 4.2. *A map of documents from F01 (Crop husbandry) subject code automatically labelled their document ids. The small white dots denote the nodes on the map of the size 36*40 nodes.*

As one can observe from the visualisation, the document ids of data vectors are distributed close to one another forming two sets of document cluster in the upper half of the map region. The location of the winning nodes for each document is indicated by their document ids across the two-dimensional map. Labels—for example 'farm management' and 'seed production'—for the corresponding document id clusters can thus be attached to characterise the two regions of the feature map.

To reinforce the results of the previous test, a second test run was conducted and this time with data vector of documents belonging to L10 subject code. This time too, the data vectors are distributed close to one another forming various clusters—one in the upper half and the other in the lower half of the map regions and a few outliers (Figure 4.3). Theses clusters could as well be conveniently labelled as 'genetics,' 'breeding,' and 'productivity.

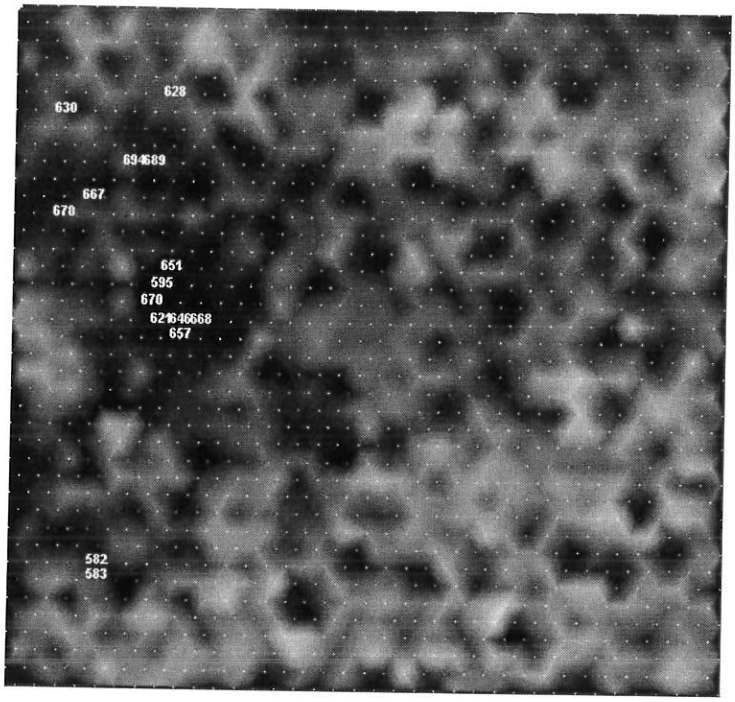


Figure 4.3. *A map of documents from L10 (Animal genetics and breeding) subject code automatically labelled their document ids. The small white dots denote the nodes on the map of the size 36*40 nodes.*

Another and more comprehensive test was finally run against the trained map using the bulk of the research publication in the collection before the entire documents were mapped. The data vector for the third test contained 640 documents (Figure 4.4). The results of this test also correspond to the two previous tests run with subsets of two different single subject code publications. Partial view of the distribution of the various publications over the different map units (nodes) (represented by the X and Y coordinate values) is presented in Table 4.8.

The clustering tendency of the map units with similar subject codes clearly proves the efficiency of the trained map and classification power of the Kohonen feature map algorithm. The result of the distribution of the entire abstracts and titles of the document collection is presented in Figure 4.5.

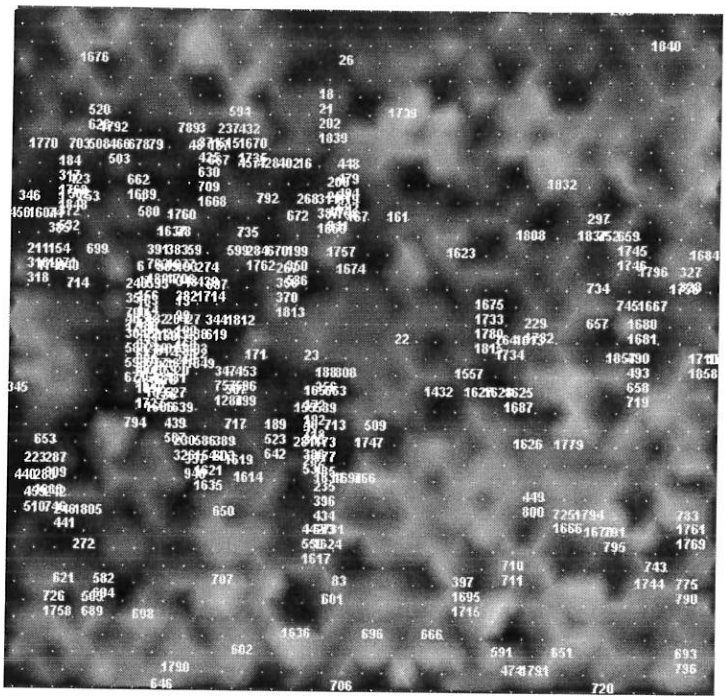


Figure 4.4. *A map of the test dataset automatically labelled using their document ids. The small white dots denote the nodes on the map of the size 36*40 nodes.*

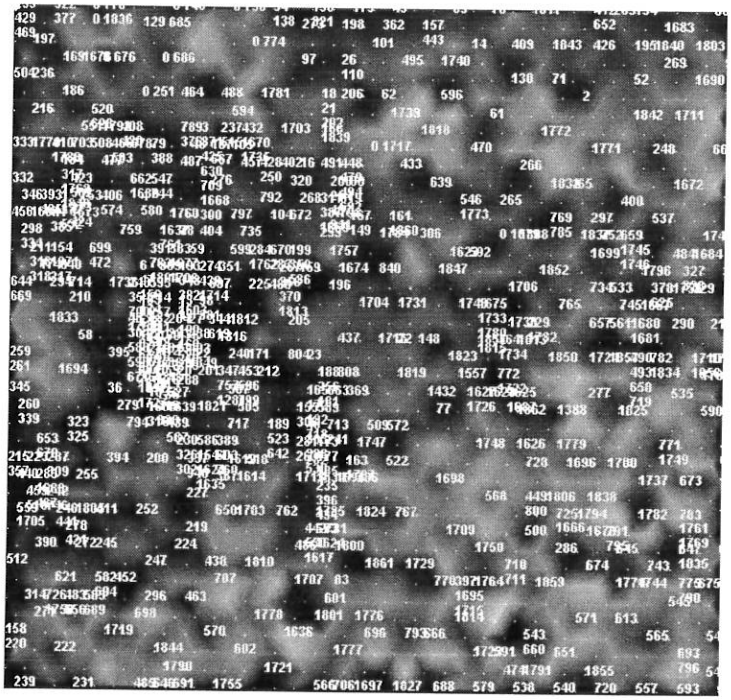


Figure 4.5. *The result of the distribution of the entire abstracts and titles of the document collection.*

A closer inspection of the table (Table 4.8), later created to keep track of the subject codes and the map unit positions and measure the efficiency of ordering of the entire document collection, is enlightening of how the map has adapted to the input data vectors.

Table 4.8. Partial view of the clustered documents and map regions on the trained map.

XCoordinate	YCoordinate	SubjectCode	Map region
0	0	L01	Top right
0	0	L01	
0	0	L30	
0	4	L02	
0	8	L10	
0	10	L10	
0	12	L02	
0	13	F30	
0	13	L30	
0	16	L02	
0	16	L10	
8	19	F70	
8	19	L73	
8	24	L73	
8	24	L73	
8	25	L72	
8	25	L72	
8	25	L72	
9	25	L73	
9	25	L73	
9	25	L73	
9	25	L73	
9	26	L73	
9	26	L73	
35	20	L02	
35	21	L02	
35	25	L02	
35	25	L02	
35	27	L02	Bottom right
35	29	L10	
35	31	L02	
35	33	L02	
35	36	L02	
35	38	L02	
35	39	L02	
39	39	L02	

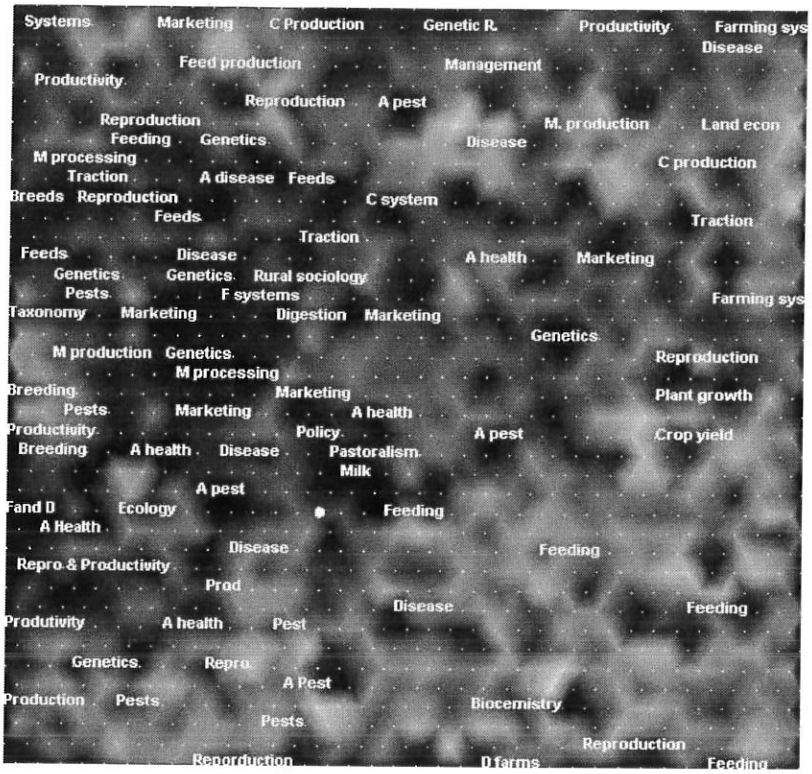
4.5 The development of the browsing interface

The other major chunk of the work in this study was the development of the browsing interface and of course the practical purpose of developing a WEBSOM map—provide an interface to a document collection so that any one could conveniently explore. This part of the

project involved creating the necessary parts for the operation of the interface (e.g., map images, html files, databases) and writing scripts to enable distributed access.

The browsing interface was developed as a series of HTML (hypertext mark-up language) pages and Java script that reads the x and y coordinates (Appendix II) of the area of the map the user clicks and passes them to an ASP (Active Server Page) (Appendix III). When a user clicks on the map area, the x and y coordinates are passed to the ASP as parameters that also computes the values of the neighbouring coordinates in all directions (left, right, up and bottom) and searches the database for matching coordinates and retrieves all associated documents with these coordinates. Matching records are then displayed to the user. An automatic method (provided by Nenet) was also used to characterise map regions. When clicking a point on the map display with a mouse, links to the document database enable reading the contents of the documents (Figure 4.6). No zooming levels, in which subsets of it can be viewed is implemented for lack of time. Only one map is used before reaching the document list and then the abstracts.

The lists of document that are retrieved would then be presented on an HTML page that contains the author, year of publications and title of the publications. Links from the titles would then enable reading the text of the abstracts.



Ridder, N. de|Wagenaar, K.T. 1986 Energy and protein balances in traditional livestock systems and ranching in eastern Botswana

Wilson, R.T. 1986 Central Mali: cattle husbandry in the agropastoral system

Armbruster, T.|Peters, K.J.|Hadji-Thomas, A. 1991 Sheep production in the humid zone of West Africa. III. Mortality and productivity of sheep in improved production systems in Cote d'Ivoire

Abstract

Compares energy and crude protein balances of cattle in traditional and ranching systems in eastern Botswana. Considers animal gross energy produced per unit, productivity at different stocking rates, liveweight, biological efficiency and pasture degradation and drought. Shows calculations of gross energy and crude protein requirements for maintaninance and production, and availability in pastures of the two systems.

Figure 4.6. *The three different view levels: the whole map, the document list and the abstract, presented in the order of increasing detail. Moving between the levels is done by mouse clicks on the image or on the document links. Once an interesting area on the map has been found, exploring the related documents in the neighbouring areas is simple. This can be contrasted with the traditional information retrieval techniques where the users cannot know whether there is a considerable number of relevant documents just outside their search results.*

4.6 Implementation of the document map

Although a complete user study has not been undertaken and is outside the scope of this project, to help capture the initial reaction of users an attempt has also been made to put the user interface to use for a few days as well as compare the retrieval results of the traditional keyword based database search and the new tool. While the initial anxiety to use the document map was quite noticeable with some users, with successive use and guidance they showed greater enthusiasm to use the map interface. Although not consistent, comparable results have also been found to the searches based on keywords. Map regions that are better focused seem to provide much better results than the traditional keyword based search. With further refinements to the labelling and better focused preprocessing parameters this result can be substantially improved.

4.7 Discussion

The document map was found to reflect relations between the documents; similar publications occur near each other on the map and are thus retrieved together. Several interesting areas containing closely related publications have been found; sample nodes from a few of them are shown in Figures 4.2, 4.3 and Table 4.8. All nodes, however, are not well focused on one subject only. While publications seem to be distributed over the entire map area, documents also overlap partially. This could be because some publications are assigned to more than one subject code while the contribution of sharing similar technical vocabularies can not be ignored, plant and animal sciences for instance. The same document may also have subsection which belong to different subject codes The visualised clustering tendency or density of the documents in different areas of the collection, presented with the colour of the document map, can be used to aid in finding related publications.

Despite the saying that no one indexer indexes the same document at two different times in a day, it can be seen that the quality of the clustering is successful for a range of subject codes. The apparent similarity of the subject codes of publication in the nearby map region

proves the quality of the mapping. The top left corner (X=0 and Y=0) of the map for instance has documents from the same subject code (L01, L02, L10 and L30). Patterns in categories L01, L02, L10 and L30 representing animal husbandry, animal feeding, animal genetics and breeding and animal ecology are also effectively grouped together. This region seems to have lamped publications from the same and more general category of animal science, production and protection. The centre left, where a good number of documents have been clustered, has documents from the same subject code L73 (Animal health). The remaining map regions also cluster other subject codes (subject code L10 (Animal genetics and breeding) lower right corner, subject code L20 (Animal ecology) and E20 (management of farms) top right corner, etc.). Other subject codes are like wise distributed in their appropriate map regions based on their similarity relations with the above subject code. The results correspond well with the accuracy in several different runs with subsets of the document collection.

Topological ordering of the map is also successful i.e., multiple subject code publications seem to have in most cases mapped to area between clusters of single subject code publications they are made of. The majority of the publications mapped to the centre of the map are assigned multiple subject codes.

The trained map thus makes successful distinction of the various types of publications in the document collection. The visualisation can help in forming a general view of the domain and it can as well guide exploration towards potential interesting particular areas. The map also provides the pattern in the type of research publications in the collection and the direction of the research undertaken by the institute. It might thus help indicate new areas of research to the institute by showing the gaps in the collection.

Finally, it is therefore quite evident the SOM has effectively adapted to the data and is thus suitable, it enables interactive browsing and exploration of the document database. Map regions are appropriately characterised with keywords, to be regarded as some kind of landmarks on the map display, to provide guidance to the exploration. These keywords serve as navigation cues during the exploration of the map, as well as provide information on the topics discussed in the documents on the respective map area. When clicking a point on the

map display with a mouse, links to the document database enable reading the contents of the documents.

Chapter Five: Conclusion and recommendation

5.1 Conclusion

This thesis was motivated by the ever increasing need to better organise a quarter of a century of accumulated knowledge and information found at ILRI. An attempt has been made to apply a novel methodology for ordering the abstracts and titles of 986 of these publications, and develop a browsing interface for exploring the resulting ordered map of the document space. The method, called WEBSOM, performs a completely automatic and unsupervised text analysis of the data vectors representing document set using the Self-organising map algorithm. The result of this work, an ordered map of the document space, displays directly the similarity relationships of the subject matter of the documents. They are reflected as distance relations on the document map. Moreover, the density of the documents in different parts of the document space can be illustrated with shades of colour on the document map display.

The results of this work also compares with the results of similar research on other collections. This work also introduced the use of an alternative ASP code to develop user interface that can be viewed using a graphical web browser while the majority of research work on WEBSOM reported in the literature actually used CGI (Common Gateway Interfaces). The document map has many kinds of potential uses in situations where previously unknown document set require managing. The map can be used for:

- Visualising and exploring an unfamiliar document collection
- Finding documents similar to a given interesting document
- Visualising the relationships between new documents and the collection
- Defining a filter for keeping, discarding or categorising new information

Exploration of the ordered map would also aid a new researcher to form an idea of the already investigated areas in livestock research. The labels guide the researcher to interesting map areas and an immediate overall idea of what kind of material the collection contains. Furthermore, when a new research publication arrives it can be placed on the map: the text of the abstracts and title is encoded and the map units most similar to it are searched for. The regions found in this way might be used as starting points for exploration, e.g, when a researcher tries to find out if there exist research publications that might overlap with the new one. It is important to note, however, that the document map is neither intended nor suitable for automatically obtaining a new categorisation for research publications. Rather, the document map offers support for exploration of the collection and for finding new, related information based on familiar information. There exists many useful visualisation of a document collection of which a single document map provides one.

To illustrate how a document map can be used to explore an unknown or partially know document collection, the entire document collection were organised onto a document map of 36 by 40 units and a browsing interface developed. The abstracts were taken from publications belonging to 82 different subject codes related to agricultural research. The vocabulary, after discarding the ratest and some of the most common words, used to encode the document collection were 421 distinct terms.

An automatic method for characterising clusters of texts and document map areas with descriptive words or labels (keywords) has been employed. The method is especially suitable for characterising maps of data collections organised using the SOM algorithm. The method focuses on the distributions of words occurring within the document groups. The method is validated using human-assigned list of descriptive terms that provide a basis for comparison.

The map implementation produced comparable results to keyword search. The efficiency of clustering in different regions varies. Some provide comparable results while others come up with fussy results. These regions seem to have been influenced by a number of non-technical terms like conduct, experiment, assess etc. justifying the need to take extra caution

in the development of stop word list and determining the threshold values to eliminate the least and most frequent terms.

It could be concluded that a large portion of the abstracts of ILRI research publications in the database deal with the more specific area of investigation reported. It could as well be observed that the average sized abstracts provide enough context for their proper placement on the map, while it might also be most appropriate to find clusters of multiple subject code publications. The contribution of the nature of the training dataset (the fact that it was entirely limited to single subject code publications) has yet to be seen. This might as well partially explain the overlap of the different subject codes on the same map region.

A clear advantage of using the same document map display for a longer time is that as the user grows familiar with the map, the display can be used as a tool for interpreting new information rapidly by overlaying the new information on top of the familiar structures. Naturally, the document collection used for organising the map must be at least somewhat similar to the incoming data stream.

Being already familiar with the document map the user may select map regions of interest to concentrate.

The region in the lower right corner contains publications on animal ecology. The larger and rather unspecified region in middle discusses a mix of subjects and which have also naturally been assigned to multiple subject codes by human indexers.

The small region on the top of the map seems to have specialised particularly on management of farms. I also monitored the classification accuracy by comparing the map with the human indexer assigned AGRIS/CARIS subject codes of the publications. The classification accuracy was compatible with the human classification adapted by the institute.

One major contribution of this thesis is that any future study related to the work performed here will benefit from the level of detail involved in the reporting, which is lacking in most of the literature on WEBSOM was a major limitation of the work. Finally, it would be helpful to let humans evaluate the results. The intuitive exploration provided by the map interface could be compared to the traditional keyword-based search.

5.2 Recommendation

The method has proven to be a useful tool in organising the text documents collection at the International Livestock Research Institute. In the present case study, the analysis of ILRI research publications, similar documents became mapped close to each other. Such order, created in an unsupervised fashion, facilitates interactive browsing and searching of related documents. The possibilities of exploratory browsing of vast sets of textual data within an interface that illustrates relations between the documents seems to be the key novel factor which is missing from current information retrieval systems.

The titles and abstracts of the publications that were analysed were mostly medium sized and contained pertinent topical information and sufficient contexts to properly represent the documents. This does not however avoid their scattering and partial overlap of nonrelated documents.

A time line for the year in which the publications were published could be an additional feature improvement for the system. Search facilities could as well be integrated to the system to indicate interesting starting areas of exploration of the map while implementing zoom in levels could not be underestimated. If time allowed comparisons with other approaches and techniques could have been attempted.

The WEBSOM method could be applied to various other collections. The natural choices for a follow up research would of course be providers of large document collections. For example news agencies, or patent offices with large achieves; who would like to open their archives online to public might use the method to provide a convenient and intuitive means of exploring the archives. Also individuals (both for personal and research reasons) will in time have picked various document databases, email messages and other text files, candidates for applying the method, to further exploring the potential of the WEBSOM in improving interactive exploration. These could be limited to the abstracts and/or the titles or based on analysis of the full texts. The efficiency of ordering of document vectors based on analysis of full-text and those limited to the abstract and/or titles may an interesting area future investigation.

With the development of a more organised and efficient indexing and stemming tools the WEBSOM method could as well be applied to languages that use scripts other that the Latin alphabet. An immediate area of research in this respect could be to pick up the pieces from both Nega's (1999) work on Amharic language stemming and Betelhem's (2002) research on n-gram based Amharic text indexing to help in the feature extraction phase. Without the availability of such tools to integrate apparent ventures in this area may involve tremendous work.

References

Belkin N.J. and Croft W.B. 1987. Retrieval Techniques. In: Williams M.E. (ed). *Annual review of information science and technology.* ??

Bethelhem Mengistu. 2002. *N-gram based automatic indexing for Amharic text.* MSc Thesis, Addis Ababa University, Addis Ababa, Ethiopia.

Chen H., Houston A.L., Sewell R.R., Schatz B.R. 1998. Internet browsing and searching: User evaluations of category map and concept space techniques. *Journal of American Society for Information Science* 49(7):582-603.

Dawit Yimam Seid. 1998. Applying interface agent technology to selective dissemination of information (SDI) user profile management: The case of ILRIAlerts. MSc Thesis, Addis Ababa University, Addis Ababa, Ethiopia.

Han J. and Kamber M. 2001. *Data mining: Concepts and techniques.* Morgan Kaufamm Publisher, Academic Press, San Diago, USA.

Harter S.P. and Hert C.A. 1997. Evaluation of information retrieval systems: Approaches, issues and methods. In: Williams M.E. (ed). *Annual review of information science and technology.* 32:3-94.

Honkela T. 1997. *Self-organising maps in natural language processing.* PhD Thesis, Helsinki University of Technology, Neural Networks Research Centre, Finland.

Honkela T., Kaski S., Lagus K. and Kohonen T. 1996. Newsgroup exploration with WEBSOM method and browsing interface. Report A32. Helsinki University of Technology, Faculty of Information Technology, Laboratory of Computer and Information Science. Helsinki, Finland.

Kaski S. 1997. *Data exploration using self-organising maps.* PhD Thesis, Helsinki University of Technology, Neural Networks Research Centre, Finish Academies Technology, Acta Polytechnica Scandinavia.

Kaski S., Honkela T., Lagus K., and Kohonen T. 1998. WEBSOM—Self-organising maps of document collections. *Neurocomputing.* 21:101-117.

Kemp D.A. 1988. *Computer-based knowledge retrieval.* Aslib, London, UK.

Kohonen T. 1998. Exploration of very large databases by self-organising maps. In: *Proceedings of ICNN'97, International Conference on Neural Networks.* pp. PL1-PL6. IEEE Service Center, Piscataway, New Jersey.

Kohonen T. 1999. Self-organisation of very large document collections: State of the art. In: Niklasson L., Doden M., and Ziemke T. (eds). *Proceedings of ICANN98, the 8th International Conference on Artificaial Neural Networks.* Vol. 1, Springer, London, UK. pp. 65-74.

Kohonen T. 2001. *Self-Organising Maps.* 3rd edition. Springer-Verlag, Berlin, Germany.

Kohonen T., Kaski S. Lagus K. and Honkela T. 1996. Very-large two-level SOM for the browsing of newsgroups. In: von der Malsburg C., von Seelen W. Vorbruggen J.C. and Sendhoff B. (eds). *Proceedings of ICANN96, International Conference on Artificial Neural Networks, Bochum, Germany, July 16-19, 1996.* Lecture Notes in Computer Science, Vol. 1112, pp. 269-274. Springer, Berlin.

Kohonen T., Kaski S., Lagus K., Salojarvi J. Paatero V. and Saarela A. 2000. Self organisation of a massive document collection. *IEEE Transactions on Neural Networks.* Special issue on Neural Networks for Data Mining and Knowledge Discovery, Vol 11, Number 3, Pages 574-585.

Kohonen T., Kaski S., Lagus K., Salojarvi J., Honkela J., Paatero V. and Saarela A. 2000. Self-organisation of a massive text document collection. *IEEE Transactions on Neural Networks, Special Issue on Neural Networks for Data Mining and Knowledge Discovery.* 11: 574-585.

Lagus K. 1997. Map of WEBSOM'97 abstracts—alternative index. In: *Proceedings of WSOM'97, Workshop on Self-organising Maps, Espoo, Finland, June 4-6.* pp. 368-372. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland.

Lagus K. 1998. Generalisability of the WEBSOM method to document collections of various types. In: *Proceedings of 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT'98).* Verlag Mainz, Aachen, Germany, Vol. 1, pp. 210-214.

Lagus K. 2000a. *Text retrieval using self-organised document maps.* Technical Report A61, Helsinki University of Technology, Laboratory of Computer and Information Science.

Lagus K. 2000b. *Text mining with WEBSOM.* PhD Thesis, Helsinki University of Technology, Neural Networks Research Centre, Finish Academies Technology, Acta Polytechnica Scandinavia, Mathematics and Computing Series No. 110.

Lagus K. and Kaski S. 1999. Keyword selection method for characterising text document maps. In: *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN'99).* IEEE Press, London, pp. 371-376.

Lagus K., Honkela T., Kaski S. and Kohonen T. 2000a. WEBSOM for textual data mining. *Artificial Intelligence Review.* 13(5/6):345-364.

Lagus K., Honkela T., Kaski S., and Kohonen T. 1996a. Self-organising maps of document collections: A new approach to interactive exploration. In: Simoudis E., Han J., and Fayyad U., (ed). *Proceedings of the second International Conference on Knowledge Discovery and Data Mining (KDD'96).* AAAI Press, Menlo Park, CA, pp. 238-243.

Lagus K., Kaski S., Honkela T., and Kohonen T. 1996b. Browsing digital libraries with the aid of self-organising maps. *Proceedings of the Fifth International World Wide Web Conference WWW5, May 6-10.* Paris, France, pp. 71-79.

Lin X. 1997. Map displays for information retrieval. *Journal of the American Society for Information Science* 48(1): 40-54.

Moore T. and Carling C. 1988. *The Limitations of Language.* Macmillan Press, Houndmills.

Nega Alemayehu. 1999. *Development of stemming algorithm for Amharic language text retrieval.* PhD Thesis, University of Sheffield, Sheffield, UK.

NNI (Neural Networks Information). 2001. http://koti.mbnet.fi/~phodju/nenet/NeuralNetworks/NeuralNetworks.html

Salton G. and McGill M.J. 1983. *Introduction to modern information retrieval.* McGraw-Hill, New York, USA.

Swanson D.R. 1988. Historical note: Information retrieval and the future of an illusion. *Journal of American Society for Information Science* 39(2):92-98.

Van Reijsbergen X. 1979. Information retrieval.

Wise J.A. 1999. The ecological approach to text visualisation. *Journal of American Society for Information Science* 50(13):1224-1233.

Appendix I

```
Function main()

Call createAbstractDataRefined

Call createtrainingdataset

Call createtestdata

Call generateword("trainingdataset", "vectortable") 'to create vector table 4 training data set

Call generateword("TestDataSet", "VectorTableTest") 'to create vector table 4 test data set

Call distinctallfromvectortable

Call computeCF("vectortable", "distinctall")

Call updatevectortablewithcf

Call CreateModifiedVectorTable

Call computetermweight("modifiedvectortable")

Call distinctallfromModifiedvectortable 'to be used as column vector headings

Call createvectorinmatrixform("modifiedvectortable", "c:\trainingdata.dat")

Call createdistinctalltest

Call distinctalltestfinal

Call computeCF("VectorTableTest", "DistinctAllTestFinal")

Call createmodifiedvectortabletest

Call computetermweight("ModifiedVectorTableTest")

Call createvectorinmatrixform("modifiedvectortabletest", "c:\testdataset1.dat")


End Function


Function createAbstractDataRefined()

'Function used to create refined data table which removes documents with no abstracts

DoCmd.RunSQL "SELECT AbstractData.ID, AbstractData.Title, AbstractData.SubjectCode,

AbstractData.Abstract INTO AbstractDataRefined " & _
```

"FROM AbstractData WHERE (((AbstractData.Abstract) Is Not Null)) ORDER BY AbstractData.ID;"

End Function

Function createtrainingdataset()

'Function used to create training dataset

On Error Resume Next

Dim rst As DAO.Recordset

Dim rst1 As DAO.Recordset

Dim count As Integer

Dim st1 As String

Dim st2 As String

st1 = "SELECT AbstractDataRefined.ID, AbstractDataRefined.Title, AbstractDataRefined.SubjectCode, AbstractDataRefined.Abstract "

st2 = " FROM AbstractDataRefined WHERE (((InStr([subjectcode], ""|"")) = ""0"")) ORDER BY AbstractDataRefined.SubjectCode"

st = st1 & st2 'training dataset are those assigned with single subject codes. Documents assigned to multiple subject codes contain a special character used to separate each subject code in the subject code field

Set rst = CurrentDb.OpenRecordset(st, dbopendynaset)

Set rst1 = CurrentDb.OpenRecordset("trainingdataset", dbopendynaset)

scode = rst!subjectcode 'take the first subject code

count = 0

While Not rst.EOF 'identify the number of records with this subject code

```
rst.FindFirst "subjectcode ='" & scode & "'"
While Not rst.NoMatch
  count = count + 1
  rst.FindNext "subjectcode ='" & scode & "'"
Wend
  p = rst.AbsolutePosition  'move to the first record with this subject code
  rst.FindFirst "subjectcode ='" & scode & "'"
  If count <> 1 Then  'if there are more than one take half of the documents with this subject
code and append them in the training table
  For i = 1 To count / 2
    rst1.AddNew
    rst1!ID = rst!ID
    rst1!title = rst!title
    rst1!subjectcode = rst!subjectcode
    rst1!Abstract = rst!Abstract
    rst.FindNext "subjectcode ='" & scode & "'"
    rst1.Update
  Next
  Else
    rst1.AddNew  'take it and append it in the training table
    rst1!ID = rst!ID
    rst1!title = rst!title
    rst1!subjectcode = rst!subjectcode
    rst1!Abstract = rst!Abstract
    rst1.Update
  End If
  count = 0
  rst.AbsolutePosition = p
```

```
    rst.MoveNext  'go to the next subject code
    scode = rst!subjectcode
Wend

End Function

Function createtestdata()
'Function used to create test dataset which are all document not considered as training data
DoCmd.RunSQL "SELECT AbstractDataRefined.ID, AbstractDataRefined.Title,
AbstractDataRefined.SubjectCode, AbstractDataRefined.Abstract INTO TestDataSet " & _
"FROM AbstractDataRefined LEFT JOIN trainingdataset ON AbstractDataRefined.ID =
trainingdataset.ID " & _
" WHERE (((trainingdataset.ID) Is Null));"

End Function

Function generateword(tabl1 As String, tabl2 As String)
'Function to create word frequency of documents
Dim rstS As DAO.Recordset 'holds the stop word list
Dim rstV As DAO.Recordset  'holds the vector table records
Dim rstT As DAO.Recordset  'holds words of a document temporarily
Dim rstD As DAO.Recordset  'holds the data set

Set rstS = CurrentDb.OpenRecordset("stopwordlist", dbopendynaset) 'holds the stopwords
Set rstT = CurrentDb.OpenRecordset("temporarytable", dbopendynaset) 'used to hold word
frequency of a document temporarily
Set rstD = CurrentDb.OpenRecordset(tabl1, dbopendynaset) 'the data table
Set rstV = CurrentDb.OpenRecordset(tabl2, dbopendynaset) 'the vector table for this data set
```

```
Dim st As String
Dim l As Integer
Dim i As Integer
Dim ch As String
Dim m As String

While Not rstD.EOF ' process each document in the data table

st = rstD!Abstract + rstD!title 'concatenate the abstract and the title of a document

l = Len(st) 'computes the length of the above text (title and abstract)

w = "" 'variable to hold each word identified
For i = 1 To l ' process each character of every word in the document
ch = Mid(st, i, 1) 'process each character
Select Case ch
Case " ", ",", "(", ")", "`", "'", """", ".", "!", "%", "&", ";", ":", "/", "-", "[", "]", "=", "+", "?", "<",
">", "*" 'set of delimiters used to identify a word
rstS.FindFirst "stopword ='" & w & "'" ' checks if the identified word is in the stop words list
If (rstS.NoMatch) And (Not IsNumeric(w)) And (w <> "") And (Len(w) > 1) Then
  m = w ' if it is not in the stop words list and is numeric and its length is greater than one it
is considered meaningful word
  w = StripSuffixes(m)  'function that stems the word identified
  w = UCase(w) 'converts all words to upper case
  rstT.FindFirst "word = '" & w & "'" ' checks if the word has already been identified for the
document
  If rstT.NoMatch Then ' if it is a new word store it in the temporary table with frequency one
```

```
  rstT.AddNew
  rstT!word = w
  rstT!frequency = 1
  rstT.Update
 Else
  rstT.Edit ' else adjust its frequency by one
  rstT!frequency = rstT!frequency + 1
  rstT.Update
 End If
End If
w = ""
Case Else
 If ch >= "0" And ch <= "9" Then ' if the character is not a word delimiter and not numeric
then concatenate it with previously taken characters
  w = ""
 Else
   w = w + ch
 End If
End Select
Next

 rstT.MoveFirst  'append identified vector of the document into the vector table and flash
from temporary table
 While Not rstT.EOF
  rstV.AddNew 'vector table
  rstV!docid = rstD!ID
  rstV!frequencyindoc = rstT!frequency
  rstV!word = rstT!word
```

```
  rstV.Update
  rstT.Delete
  If Not rstT.BOF Then rstT.MoveFirst
Wend

rstD.MoveNext  'move to the next document
Wend

End Function

Function distinctallfromvectortable()
'Function used to identify distinct terms from the identified vector table
DoCmd.RunSQL "SELECT DISTINCT VectorTable.word,  0 AS DF INTO DistinctAll FROM
VectorTable ORDER BY VectorTable.word;"

End Function

Function computeCF(tabl1 As String, tabl2 As String)
'Function to compute collection frequency of words identified in the function generate word
Dim rst1 As DAO.Recordset
Dim rst2 As DAO.Recordset
Set rst1 = CurrentDb.OpenRecordset(tabl1, dbopendynaset) 'table1 is data set
Set rst2 = CurrentDb.OpenRecordset(tabl2, dbopendynaset) 'holds distinct all terms from
vector table
While Not rst2.EOF  'processes collection frequency for each distinct term from vector table
count = 0
rst1.FindFirst "word ='" & rst2!word & "'"
```

```
While Not rst1.EOF And Not rst1.NoMatch 'searches documents with this word in the vector
table
count = count + 1
rst1.FindNext "word ='" & rst2!word & "'"
Wend
rst2.Edit 'updates collection frequency in the distinct all terms table
rst2!DF = count
rst2.Update
rst2.MoveNext 'move to the next term and process collection frequency
Wend

End Function

Function updatevectortablewithcf()
'Function used to update collection frequency of terms in the vector table from distinct all
table
DoCmd.RunSQL "UPDATE vectortable INNER JOIN DistinctAll ON
[vectortable].[word]=[DistinctAll].[word] SET vectortable.DF = [distinctall]![df];"
End Function

Function CreateModifiedVectorTable()
'Function used to create modified vector table which holds document id, word and each term
frequency from vector table and the terms collection frequency from distinct all table. Uses
threshold values to remove those terms above and below the threshold values
DoCmd.RunSQL "SELECT VectorTable.docid, VectorTable.word,
VectorTable.frequencyindoc, " & _
"VectorTable.DF, 0 AS Weight INTO ModifiedVectorTable FROM VectorTable " & _
```

```
"WHERE (((VectorTable.DF) Between [Enter the minimum CF value below which a
document should be avoided] " & _
"And [Enter the maximum CF value below which a document should be avoided])) ORDER
BY vectortable.docid, vectortable.word;"

End Function

Function computetermweight(tabl As String)
'Function used to compute term weight for modified vector table

Dim rst1 As DAO.Recordset
Dim nodoc As Integer

Set rst1 = CurrentDb.OpenRecordset(tabl, dbopendynaset)

nodoc = InputBox("Enter the number of documents", "Data Entry Form")

While Not rst1.EOF 'compute the weight for each term in the table
  rst1.Edit
  rst1!Weight = rst1!DF * Log(nodoc / rst1!frequencyindoc)
  rst1.Update
rst1.MoveNext
Wend

End Function

Function distinctallfromModifiedvectortable()
'Function used to create a table which holds distinct terms from modified vector table
```

```
'Function to create a table which holds distinct terms from modified vector table
DoCmd.RunSQL "SELECT Distinct ModifiedVectorTable.word INTO DistinctAllFromModifiedVectorTable FROM ModifiedVectorTable order by ModifiedVectorTable.word;"

End Function

Function createvectorinmatrixform(tabl1 As String, fname As String)
'Function used to create the data vector in matrix form in a text file. Column heading will be distinct words and row headings will be document ids
Dim rst As DAO.Recordset
Dim rst1 As DAO.Recordset
Set rst = CurrentDb.OpenRecordset(tabl1, dbopendynaset) 'tabl1 is modified vector table
Set rst1 = CurrentDb.OpenRecordset("DistinctAllFromModifiedVectorTable", dbopendynaset)' 'distinct terms from modified vector table
Set fs = CreateObject("Scripting.FileSystemObject") 'creates file system object
Set f = fs.createtextfile(fname, True) 'open the file
rst1.MoveLast
d = rst1.RecordCount 'count the number of distinct words which is the vector size
On Error GoTo last 'error handler
f.WriteLine (d) 'write the dimension of the vector first
ID = rst!docid
While Not rst.EOF 'process for each record
j = 1
   While ID = rst!docid 'process for each document

      w = rst!word 'take each word of the document under consideration
```

```
    rst1.FindFirst "word ='" & w & "'"   'identify its postion in the distinct all from modified
vector table which tells on which column its value should be posted
    p = rst1.AbsolutePosition + 1
    If p > 1 Then  'identify how far it should be placed from the current cursor position
      For i = j To p - 1
        f.Write ("0 ")  'post zero values up to that column position in the file
      Next
      j = i + 1
      f.Write (rst!Weight & " ")  'write the weight of the term at its column
     Else
      f.Write (rst!Weight & " ")  'write the weight at the current position
     End If
     rst.MoveNext
   Wend

    For i = 1 To d – p  'if not other term exist for this document and the end of the column has
not been reached, write zero values to all the remaining columns
       f.Write ("0 ")
    Next
    f.WriteLine (ID)  'append the document ids at the end to serve as labels
    ID = rst!docid
  Wend
  last:
  f.Close  'close the file

  End Function

  Function createdistinctalltest()
```

```
'Function that creates distinct term from the test data set which may contain terms not found
in the real vector in the text file created for training vector
DoCmd.RunSQL "SELECT vectortabletest.word, 0 AS DF INTO DistinctAlltest FROM
vectortabletest GROUP BY vectortabletest.word, 0;"

End Function

Function distinctalltestfinal()
'Function used to create distinct terms which are contained in real vector file
DoCmd.RunSQL "SELECT DistinctAllTest.word, 0 AS DF INTO DistinctAllTestFinal " & _
"FROM DistinctAllFromModifiedVectorTable INNER JOIN DistinctAllTest ON
DistinctAllFromModifiedVectorTable.word = DistinctAllTest.word " & _
"ORDER BY DistinctAllTest.word;"

End Function

Function createmodifiedvectortabletest()
'Function used to create modified vector table for test data set
DoCmd.RunSQL "SELECT VectorTableTest.docid, VectorTableTest.word,
VectorTableTest.frequencyindoc, DistinctAllTestFinal.DF, 0 AS Weight INTO
ModifiedVectorTableTest " & _
"FROM VectorTableTest INNER JOIN DistinctAllTestFinal ON VectorTableTest.word =
DistinctAllTestFinal.word " & _
"WHERE (((DistinctAllTestFinal.DF) Between [Enter Minimum CF] And [Enter Maximum
CF])) " & _
"ORDER BY VectorTableTest.docid, VectorTableTest.word;"

End Function
```

```
Option Compare Database
'Global variables used by the various functions of the stemmer
Option Base 1
Dim suffixes2(22, 2) As String
Dim suffixes3(8, 2) As String
Dim suffixes4(21) As String
Dim initialized As Boolean

'A subroutine to initialise prefixes
 Public Sub InitConflater()
 Dim stopwords(35) As String
 Dim prefixes(10) As String
 prefixes(1) = "kilo"
 prefixes(2) = "micro"
 prefixes(3) = "milli"
 prefixes(4) = "intra"
 prefixes(5) = "ultra"
 prefixes(6) = "mega"
 prefixes(7) = "nano"
 prefixes(8) = "pico"
 prefixes(9) = "pseudo"
 End Sub

 Public Sub ToLowerCase(kwd As String)
 ToLowerCase = LCase(kwd)
 End Sub
```

```
'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
Public Function IsValid(l)
'A function used to determine the validity of a word. A word is valid if it does not contain
alphanumeric characters
If InStr(1, "abcdefghijklmnopqrstuvwxyz", l) <> 0 Then
IsValid = True
ElseIf InStr(1, "ABCDEFGHIJKLMNOPQRSTUVWXYZ", l) <> 0 Then
IsValid = True
ElseIf InStr(1, "0123456789", l) <> 0 Then
IsValid = True
Else
IsValid = False
End If
End Function

'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ

Public Function Clean(kwd As String)
Dim newKwd As String
Dim i As Integer

newKwd = ""
For i = 1 To Len(kwd)
If IsValid(Mid(kwd, i, 1)) Then
newKwd = newKwd & Mid(kwd, i, 1)
End If
Next
Clean = newKwd 'WATCH
```

```
End Function

'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
Public Function StopWordTest(bottom, top As Integer, word As String)
'Function used to check if a word is a stop word or not.
Dim mmid As Integer
Dim midWord As String

If bottom > top Then
StopWordTest = False
Else
mmid = ((top - bottom) / 2) + bottom
midWord = stopwords(mmid)
If word < midWord Then
StopWordTest = StopWordTest(bottom, mmid - 1, word)
ElseIf word > midWord Then
StopWordTest = StopWordTest(mmid + 1, top, word)
Else
StopWordTest = True
End If
End If
End Function

'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ}
Public Function StripPrefixes(strng As String)
'Function used to strip off prefixes from a word
Dim i As Integer
For i = 1 To 9
```

```
If Mid(strng, 1, Len(prefixes(i))) = prefixes(i) Then
strng = Mid(strng, Len(prefixes(i)) + 1, Len(strng) - Len(prefixes(i)))
GoTo last
End If
Next
last:
StripPrefixes = strng
End Function


'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ}
Public Function HasSuffix(word As String, suffix As String, stem As _
'Function used to determine if a word has known prefixes
String) As Boolean

If Len(word) <= Len(suffix) Then
HasSuffix = False
GoTo last
End If
If Len(suffix) > 1 Then
If Mid(word, (Len(word) - 1), 1) <> Mid(suffix, Len(suffix) - 1, 1) Then
HasSuffix = False
GoTo last
End If
End If
stem = Mid(word, 1, Len(word) - Len(suffix))
If stem & suffix = word Then
HasSuffix = True
```

```
 Else
 HasSuffix = False
 End If
last:
 End Function

'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ}
 Public Function Vowel(ch, prev)
 'Function used to determine if a character in a word is vowel or not
Select Case ch
 Case "a", "e", "i", "o", "u":
 Vowel = True
 Case "y":
 Vowel = (Vowel(prev, "?"))
 Case Else
 Vowel = False
 End Select
 End Function

'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ}
 Public Function Cvc(strng As String)
 'Function used to determine the pattern of a sequence of characters
Dim lenn As Integer
 Dim l As String

 lenn = Len(strng)
```

```
If lenn < 3 Then
Cvc = False
ElseIf (Vowel(Mid(strng, lenn, 1), Mid(strng, lenn - 1, 1)) = _
False) And (InStr("wxy", Mid(strng, lenn, 1) <> 0) And _
(Vowel(Mid(strng, lenn - 1, 1), Mid(strng, lenn - 2, 1)) = True)) _
Then
If lenn = 3 Then
If Vowel(Mid(strng, 1, 1), "?") = False Then
Cvc = True
Else
Cvc = False
End If
Else
If Vowel(Mid(strng, lenn - 2, 1), Mid(strng, lenn - 3, 1)) = False Then
Cvc = True
Else
Cvc = False
End If
End If
Else
Cvc = False
End If
End Function

'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ}

Public Function Measure(stem As String)
Dim i, count, lenn
```

```
lenn = Len(stem)
count = 0
i = 1
Do While i <= lenn

Do While i <= lenn
If i > 1 Then
If Vowel(Mid(stem, i, 1), Mid(stem, i - 1, 1)) = True Then
GoTo l1
End If
ElseIf Vowel(Mid(stem, i, 1), "?") = True Then
GoTo l1
End If
i = i + 1
Loop
l1:
i = i + 1
Do While i <= lenn
If i > 1 Then
If Vowel(Mid(stem, i, 1), Mid(stem, i - 1, 1)) = False Then
GoTo l2
End If
ElseIf Vowel(Mid(stem, i, 1), "?") = False Then
GoTo l2
End If
i = i + 1
Loop
```

```
l2:
If i <= lenn Then
count = count + 1
i = i + 1
End If
Loop
Measure = count
End Function

'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff}
Public Function ContainsVowel(word As String)
'Function used to determine if a word contains vowels inside
Dim i As Integer
Dim sstop As Boolean

sstop = False
i = 1
Do While (Not sstop) And (i <= Len(word))
If i > 1 Then
If Vowel(Mid(word, i, 1), Mid(word, i - 1, 1)) = True Then
sstop = True
End If
ElseIf Vowel(Mid(word, i, 1), "?") = True Then
sstop = True
End If
i = i + 1
Loop
ContainsVowel = sstop
```

```
End Function

'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ}
Public Function Step1(strng As String)
'Function that actually strips off the first set of suffixes
Dim stem As String
Dim lenn As Integer

If Mid(strng, Len(strng), 1) = "s" Then
If (HasSuffix(strng, "sses", stem) = True) Then
strng = Mid(strng, 1, Len(strng) - 2)
ElseIf (HasSuffix(strng, "ies", stem) = True) Then
strng = Mid(strng, 1, Len(strng) - 2)
ElseIf Mid(strng, Len(strng) - 1, 1) <> "s" Then
'if its an 's' then lob if off}
strng = Mid(strng, 1, Len(strng) - 1)
End If
End If

If HasSuffix(strng, "eed", stem) = True Then
If Measure(stem) > 0 Then
strng = Mid(strng, 1, Len(strng) - 1)
End If
ElseIf ((HasSuffix(strng, "ed", stem) = True) Or (HasSuffix(strng, _
"ing", stem) = True)) And (ContainsVowel(stem) = True) Then
strng = Mid(strng, 1, Len(stem))
If (HasSuffix(strng, "at", stem) = True) Or (HasSuffix(strng, "bl", _
stem) = True) Or (HasSuffix(strng, "iz", stem) = True) Then
```

```
  strng = strng & "e"
  Else
  lenn = Len(strng)
  If lenn > 1 Then
  If (Mid(strng, lenn, 1) = Mid(strng, lenn - 1, 1)) And (Mid(strng, _
lenn, 1) <> "l") And (Mid(strng, lenn, 1) <> "s") And (Mid(strng, _
lenn, 1) <> "z") Then
  strng = Mid(strng, 1, lenn - 1)
  End If
  ElseIf Measure(strng) = 1 Then
  If Cvc(strng) = True Then
  strng = strng & "e"
  End If
  End If
  End If
  End If
  'End of Step 1b

  'Beggining of Step 1c

  ' we made this correction to be reviewed later
' If HasSuffix(strng, "y", stem) Then
' If ContainsVowel(stem) Then
' tmp_strng = Mid(strng, 1, Len(strng) - 1)
' tmp_strng = tmp_strng & "i"
' strng = tmp_strng
' End If
' End If
```

```
Step1 = strng
'End of Step 1c
End Function

'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ}

Public Function Step2(strng As String)
'Function that actually strips off the second set of suffixes
Dim stem As String
Dim index As Integer

For index = 1 To 22
If HasSuffix(strng, suffixes2(index, 1), stem) = True Then
If Measure(stem) > 0 Then
strng = stem & suffixes2(index, 2)
GoTo l1
End If
End If
Next
l1:
Step2 = strng
End Function

'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ}

Public Function Step3(strng As String)
'Function that actually strips off the third set of suffixes
Dim stem As String
```

```
Dim index As Integer

For index = 1 To 8
If HasSuffix(strng, suffixes3(index, 1), stem) = True Then
If Measure(stem) > 0 Then
strng = stem & suffixes3(index, 2)
GoTo l1
End If
End If
Next
l1:
Step3 = strng
End Function

'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ}

Public Function Step4(strng As String)
'Function that actually strips off the fourth set of suffixes
Dim index As Integer
Dim stem As String

For index = 1 To 21
If HasSuffix(strng, suffixes4(index), stem) = True Then
If Measure(stem) > 1 Then
strng = stem
'GoTo l1
Exit For
End If
```

```
End If
Next
l1:
Step4 = strng
End Function


'ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff}


Public Function Step5(strng As String)
'Function that actually strips off the fifth set of suffixes
Dim stem As String


'Beggining of Step 5a}
'If strng <> "" Then


If Mid(strng, Len(strng), 1) = "e" Then
If Measure(strng) > 1 Then
strng = Mid(strng, 1, Len(strng) - 1)
ElseIf Measure(strng) = 1 Then
stem = Mid(strng, 1, Len(strng) - 1)
If Cvc(stem) = False Then
strng = Mid(strng, 1, Len(strng) - 1)
End If
End If
End If


'End If
```

```
 'End of Step 5a}

 'Beggining of Step 5b}
 'If strng <> "" Then
If Len(strng) = 1 Then GoTo l1
 If (Mid(strng, Len(strng), 1) = "l") And (Mid(strng, Len(strng) - _
1, 1) = "l") And (Measure(strng) > 1) Then
 strng = Mid(strng, 1, Len(strng) - 1)
 End If
 'End If
 'End of Step 5b}
l1:  Step5 = strng
 End Function

Sub initialize()
'Subroutine to initialise the suffix array
 suffixes2(1, 1) = "ational"
 suffixes2(1, 2) = "ate"
 suffixes2(2, 1) = "tional"
 suffixes2(2, 2) = "tion"
 suffixes2(3, 1) = "enci"
 suffixes2(3, 2) = "ence"
 suffixes2(4, 1) = "anci"
 suffixes2(4, 2) = "ance"
 suffixes2(5, 1) = "izer"
 suffixes2(5, 2) = "ize"
 suffixes2(6, 1) = "iser"
 suffixes2(6, 2) = "ize"
```

suffixes2(7, 1) = "abli"

suffixes2(7, 2) = "able"

suffixes2(8, 1) = "alli"

suffixes2(8, 2) = "al"

suffixes2(9, 1) = "entli"

suffixes2(9, 2) = "ent"

suffixes2(10, 1) = "eli"

suffixes2(10, 2) = "e"

suffixes2(11, 1) = "ousli"

suffixes2(11, 2) = "ous"

suffixes2(12, 1) = "ization"

suffixes2(12, 2) = "ize"

suffixes2(13, 1) = "isation"

suffixes2(13, 2) = "ize"

suffixes2(14, 1) = "ation"

suffixes2(14, 2) = "ate"

suffixes2(15, 1) = "ator"

suffixes2(15, 2) = "ate"

suffixes2(16, 1) = "alism"

suffixes2(16, 2) = "al"

suffixes2(17, 1) = "iveness"

suffixes2(17, 2) = "ive"

suffixes2(18, 1) = "fulness"

suffixes2(18, 2) = "ful"

suffixes2(19, 1) = "ousness"

suffixes2(19, 2) = "ous"

suffixes2(20, 1) = "aliti"

suffixes2(20, 2) = "al"

suffixes2(21, 1) = "iviti"

suffixes2(21, 2) = "ive"

suffixes2(22, 1) = "biliti"

suffixes2(22, 2) = "ble"

suffixes3(1, 1) = "icate"

suffixes3(1, 2) = "ic"

suffixes3(2, 1) = "ative"

suffixes3(2, 2) = ""

suffixes3(3, 1) = "alize"

suffixes3(3, 2) = "al"

suffixes3(4, 1) = "alise"

suffixes3(4, 2) = "al"

suffixes3(5, 1) = "iciti"

suffixes3(5, 2) = "ic"

suffixes3(6, 1) = "ical"

suffixes3(6, 2) = "ic"

suffixes3(7, 1) = "ful"

suffixes3(7, 2) = ""

suffixes3(8, 1) = "ness"

suffixes3(8, 2) = ""

suffixes4(1) = "al"

suffixes4(2) = "ance"

suffixes4(3) = "ence"

suffixes4(4) = "er"

suffixes4(5) = "ic"

suffixes4(6) = "able"

```
suffixes4(7) = "ible"
suffixes4(8) = "ant"
suffixes4(9) = "ement"
suffixes4(10) = "ment"
suffixes4(11) = "ent"
suffixes4(12) = "sion"
suffixes4(13) = "tion"
suffixes4(14) = "ou"
suffixes4(15) = "ism"
suffixes4(16) = "ate"
suffixes4(17) = "iti"
suffixes4(18) = "ous"
suffixes4(19) = "ive"
suffixes4(20) = "ize"
suffixes4(21) = "ise"

End Sub

'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ}
Public Function StripSuffixes(strng As String)
'Main stemmer function which call the associated functions
If Not initialized Then
  Call initialize
  initialized = True
End If
strng = Step1(strng)
strng = Step2(strng)
strng = Step3(strng)
```

```
strng = Step4(strng)

strng = Step5(strng)

StripSuffixes = strng


End Function



'ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ}

Public Function StripAffixes(strng As String)

Function to remove affixes from words

strng = LCase(strng)

strng = Clean(strng)

If (strng <> "") And (Len(strng) > 2) Then

If Not (StopWordTest(1, 319, strng)) Then

strng = StripPrefixes(strng)

If strng <> "" Then

strng = StripSuffixes(strng)

End If

Else

strng = ""

End If

Else

strng = ""

End If

StripAffixes = strng

End Function


Function stemtheword(x)
```

```
End Function
```

Appendix II

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
        <title>X, Y Coordinate</title>
</head>

<body>
<SCRIPT LANGUAGE="JavaScript">
<!--
 function coordinate(a,b) {
        document.write("<form action='fetch.asp' method='POST' name='myForm'>");
        document.write("<input type='hidden' name='x_co' value=''>");
        document.write("<input type='hidden' name='y_co' value=''>");
        document.write("</form>");

        myForm.x_co.value = a;
        myForm.y_co.value = b;

   myForm.submit ();
 }
//-->

</script>
<!--
<SCRIPT LANGUAGE="JavaScript">
```

```
<!--

 function next(x,y) {
 var c=x,d=y;
        //coordinate(c,d);

}
//-->
<!--
</SCRIPT>
-->
<A HREF="mulugeta.htm" onclick="coordinate()"><IMG SRC="map.JPG" alt="" border=0
ismap width="640" height="480"></A>

<SCRIPT LANGUAGE=JAVASCRIPT>
<!--

  if (location.search.length > 0) {
  var str = location.search;
  var commaloc = str.indexOf(",");
  coordinate(str.substring(1, commaloc),str.substring(commaloc+1,str.length));

}
//-->
</SCRIPT>

</body>
</html>
```

Appendix III

```
<%@ language="VBScript" %>
<%
Option Explicit
Response.Expires = 0
%>
<!--#include file="adovbs.inc"-->

<%
Dim conn, strconn, rs, qry
Dim x, y, desc, descrip
Dim i, j, Flag
Dim arr_x (), arr_y ()
Dim titl, autr, aut, abs, abst, yr, k, m
Redim arr_x (1090)
Redim arr_y (1090)

x = request.form("x_co")
y = request.form("y_co")

Flag = True
j=1
'arr_x(0) = x
'arr_y(0) = y
m=1
if (x >="0" and x <= "576") and (y >= "0" and y <= "1440") then
  if (x-16 >= 0) and (x+16 <= 576) and (y-16 >= 0) and (y+16 <= 1440) then
```

```
 for i=y-16 to y+16
  for j=x-16 to x+16
   arr_x(m) = j
   arr_y(m) = i
   m = m+1
  next
 next
elseif (x-16 < 0) and (y-16 >= 0) and (y+16 <= 1440) then
 for i=y-16 to y+16
  for j=0 to x+16
   arr_x(m) = j
   arr_y(m) = i
   m = m+1
  next
 next
elseif (x-16 < 0) and (y-16 < 0) and (y+16 <= 1440) then
 for i=0 to y+16
  for j=0 to x+16
   arr_x(m) = j
   arr_y(m) = i
   m = m+1
  next
 next
elseif (x-16 < 0) and (y+16 > 1440) and (y-16 >= 0) then
 for i=y-16 to 1440
  for j=0 to x+16
   arr_x(m) = j
   arr_y(m) = i
```

```
    m = m+1
   next
  next
 elseif (x+16 > 576) and (y-16 < 0) and (y+16 <= 1440) then
  for i=0 to y+16
   for j=x-16 to 576
    arr_x(m) = j
    arr_y(m) = i
    m = m+1
   next
  next
 elseif (x+16 > 576) and (y-16 >= 0) and (y+16 <= 1440) then
  for i=y-16 to y+16
   for j=x-16 to 576
    arr_x(m) = j
    arr_y(m) = i
    m = m+1
   next
  next
 elseif (x+16 > 576) and (y+16 > 1440) and (x-16 >= 0) then
  for i=y-16 to 1440
   for j=x-16 to 576
    arr_x(m) = j
    arr_y(m) = i
    m = m+1
   next
  next
 elseif (x-16 >= 0) and (x+16 <=576) and (y+16 > 1440) then
```

```
 for i=y-16 to 1400
  for j=x-16 to x+16
   arr_x(m) = j
   arr_y(m) = i
   m = m+1
  next
 next
elseif (x-16 >= 0) and (x+16 <= 576) and (y-16 < 0) then
 for i=0 to y+16
  for j=x-16 to x+16
   arr_x(m) = j
   arr_y(m) = i
   m = m+1
  next
 next
 end if
else
 Flag = False
end if

set conn = server.createobject("ADODB.connection")
strconn= "data Source =inmagic; UID=sa;Password=;"
conn.open strconn

set rs = server.createobject("ADODB.recordset")
set rs.activeconnection = conn

rs.cursorlocation = aduseserver
```

```
rs.cursortype = adopenstatic

k = 0
if Flag then
for i=LBound(arr_x) to m-1
        qry = "SELECT * FROM ILRIPub WHERE xcoordinate='" & arr_x (i) & "' and
ycoordinate ='" & arr_y (i) & "'"
        rs.open qry

 if rs.recordcount >= 1 then
  k = k+1
       Do while not rs.EOF
        titl = rs.fields("Title").value
        autr = rs.fields("Author").value
        abst = rs.fields("Abstract").value
        yr = rs.fields("yearofpublication").value
        response.write(autr) & "  "
        response.write(yr) & "  "
        response.write "<a href='description.asp?abs=" & abst & "'>" & titl & "</a><br>"

        %>
        <!--
        <a href="description.asp?abs=<% =abst %>&aut=<% =autr %>"><% =titl
%></a><p>
  -->
        <%
        rs.movenext
       loop
```

```
  else
    Flag = False
  end if
  rs.close
next
else
   response.write("Sorry, no match is found. Please try again.")
   Flag = True
end if

if (not flag) and (k=0) then
  response.write("Sorry, no match is found. Please try again.")
end if

set strconn = nothing
set conn = nothing

%>
```

Declaration

This thesis is my original work and has not been submitted for a degree in any other university and all sources of material used for the thesis have been duly acknowledged.

Mulugeta Bayeh

Mulugeta Bayeh Taye
June 2002

This thesis has been submitted for examination with our approval as university advisors.

Dereje Teferi

June 2002

Saba Amsalu

June 2002