Intranet indexing using semantic document clustering

Master of Science Thesis

Peter Ljungstrand & Henrik Johansson
Department of Informatics
Göteborg University
Box 620, 40530 Göteborg, Sweden
{s94pelj, s94henri}@student.informatics.gu.se

May 1998

Abstract

This thesis presents a system that applies automatic indexing techniques to large, dynamic, networked information collections, and which has been applied to a corporate intranet. The structure of the intranet is sought and an attempt is made to explore the underlying semantics of the intranet's constituent documents in order to provide an overview. An important objective is to facilitate easier navigation than is possible today. We propose a system that creates a hierarchical index and which allows for browsing at different granularity levels. The main focus is however on the indexing techniques, and most of the work is based on the theory of Information Retrieval. A prototype has been implemented and evaluated, and we conclude that the techniques applied are valuable and usable for the proposed domain.

Intranet indexering med semantisk dokumentklustring

Magisteruppsats 20p

Peter Ljungstrand & Henrik Johansson

Institutionen för Informatik

Göteborgs Universitet

Box 620, 40530 Göteborg

{s94pelj, s94henri}@student.informatik.gu.se

Maj 1998

Sammanfattning

Denna uppsats presenterar ett system baserat på automatiska indexeringstekniker avsett för stora, dynamiska och nätverksbaserade informationssamlingar, tillämpat på ett företags intranet. Syftet är att beskriva dess struktur, baserat på dokumentens innehåll och semantik, för att möjliggöra en överblick av innehållet. Ett användningsområde är att underlätta navigering i intranet. Vi föreslår ett system som skapar ett hierarkiskt index som är möjliggör ’surfning’ i strukturen. Större delen av uppsatsen inriktas på indexeringstekniker, varav de flesta härstammar från forskning inom Information Retrieval. Vi har utvecklat en prototyp varvid vi använt oss av en iterativ utvecklingsmetod. Slutligen drar vi slutsatsen att de föreslagna teknikerna är användbara för automatisk indexering och kan nyttjas för att få den överblick som söks.

1 INTRODUCTION 5
1.1 BACKGROUND 6
1.2 DISPOSITION 6
1.3 ACKNOWLEDGEMENTS 7
2 PROBLEM DESCRIPTION 8
2.1 PROBLEM DEFINITION 8
2.2 PROBLEM DECOMPOSITION 8
3 METHOD 10
3.1 LITERATURE REVIEW 10
3.1.1 Literature gathering 10
3.1.2 Literature analysis 11
3.2 SEARCH FOR SUITABLE SOFTWARE 11
3.3 PROTOTYPING 12
4 BACKGROUND 14
4.1 INFORMATION OVERLOAD 14
4.2 CONCEPTUAL FRAMEWORK OF INFORMATION SEEKING 14
4.3 NAVIGATION AIDS 15
4.4 INTRANETS 16
4.5 VOLVO’S INTRANET 17
5 TECHNIQUES AND ALGORITHMS 18
5.1 AUTOMATIC TEXT ANALYSIS 18
5.2 INFORMATION RETRIEVAL FROM THE WEB 19
5.2.1 Libraries versus the Web 19
5.2.2 Web robots 20
5.3 DEFINITIONS OF INFORMATION RETRIEVAL 20
5.4 TRADITIONAL INFORMATION RETRIEVAL 21
5.4.1 Information Retrieval Models 22
5.4.2 The Origin of Vector Space Models 22
5.4.3 The Vector Space Model 22
5.5 TERM WEIGHTING 24
5.5.1 Global Weighting 25
5.6 FEATURE SELECTION AND DIMENSIONALITY REDUCTION 25
5.6.1 Stemming 25
5.6.2 Stop Words 26
5.6.3 Reduced-Space Models 26
5.7 MEASUREMENTS IN IR SYSTEMS 26
5.7.1 Measures of association 26
5.7.2 Evaluation performance of IR systems 27
5.8 LATENT SEMANTIC INDEXING 27
5.8.1 Pros and cons with LSI 28
5.9 CLUSTERING TECHNIQUES 29
6 RESULTS 32
6.1 CONCEPTUAL DESIGN IMPLICATIONS 32
6.2 TECHNICAL DESCRIPTION OF OUR SYSTEM 32
6.2.1 The Harvest system 33
6.2.2 Parsing the Harvest index structure 33
6.2.3 Stripping non-alphabetical characters 33

6.2.4 Creating the word list ... 34
6.2.5 Stop-word filtering ... 34
6.2.6 Stemming ... 36
6.2.7 Further refinement ... 37
6.2.8 Onto the documents themselves ... 37
6.2.9 Building the matrix ... 37
6.2.10 Singular Value Decomposition and LSI ... 37
6.2.11 Clustering ... 38
6.2.12 Browsing the clustered index ... 38
6.3 IMPLEMENTATION DETAILS ... 39
6.4 EVALUATION ... 39
7 DISCUSSION ... 40
7.1 FUTURE WORK ... 40
7.2 CONCLUSIONS ... 40
8 APPENDICES ... 42
8.1 APPENDIX A – MATHEMATICAL OVERVIEW OF LSI ... 42
8.2 APPENDIX B – COMMON SIMILARITY MEASURES ... 44
8.3 APPENDIX C – COMMON WEIGHTING FUNCTIONS ... 45
8.4 APPENDIX D – HARVEST OVERVIEW ... 46
8.4.1 The SMART Information Retrieval System ... 47
8.4.2 Self-Organizing Maps ... 47
8.4.3 Scatter/Gather ... 49
8.4.4 Agent-based approaches ... 51
9 BIBLIOGRAPHY ... 53

1 Introduction

In today’s advanced world, with an ever increasing amount of information available, dynamically changing in structure, content and context at a rate none would have thought of just a couple of years ago, the need to find the *right* information in this flow is crucial for success. This phenomenon is known as the *information overload* problem (Nelson 1994), already recognized in 1945 by Vannevar Bush (Bush 1945). But how do we handle this? We cannot give an exact answer to that, but we will present a tool that provides users of a large corporate intranet with assistance. Though, this system will not provide a total solution to the problem, it is rather a complement to existing standard techniques for intranet guiding, such as search engines and manually maintained indexes. This approach can easily be incorporated with other new evolving techniques, such as recommender systems and personal software agents, to accomplish even better results.

The main idea in our work is to provide a way for the intranet user to get a good overview of the content and structure of the entire intranet, with zooming possibilities. This opportunity has been lacking until now. One can compare our system to a manually maintained topic-based index, such as Yahoo or Volvo IntraPages, but with a very important difference - our system is fully automatic (unsupervised), meaning web publishers (organizations, individuals or programs) do not have to report when changes are made. The system even adapts itself to any new topics or contexts that may appear within the intranet collection. In the development of this system we have used ideas and methods from many disciplines. This has resulted in a wide theoretical basis, trying to combine results from a broad spectrum of research areas, such as information retrieval, linguistics, mathematics and computer science. To our knowledge, there exists no other system today trying to solve the problem as we do.

The method we present here is not yet mature, i.e. it is still in its development stage and should not be considered as an ultimate solution. We are not taking into account all available document attributes and file types. In order to try out our ideas through all different steps, using limited resources such as time and computational power, we have been forced to make some simplifications of the problem domain. We do not consider all possible file formats and we also disregard some meta information such as HTML tags. Many extensions and additions to the method may be applied later on to try to improve the results further.

If we in a few sentences should explain how our system works, it uses a web indexing tool (web robot or spider) that wanderers the intranet and builds an index structure. The documents found are then analyzed and clustered. Our first approach is to perform a linguistic analysis to find the most important words in the entire document corpus and give them different weighting. Using the outcome of this analysis, the next step is to create a mathematical high-dimensional model of all documents and their inherent inter-relationships. This model is further refined using linear algebra technique and results in a compact and efficient way to describe the implicit semantic inter-relationships within the document collection. To proceed from here, we create a hierarchical tree structure of the documents, using mathematical clustering techniques. Finally, the user is presented with an overview of the entire intranet. Starting from the root node of the tree, the closest branches are assembled into cluster units, described by a few parameters: typical keywords, typical documents, cluster size and depth. All this information is contained on a single screen, making it a quite easy task to select which clusters are interesting and which are not. One cluster is selected and is used as the new root node to provide further details about it, in the same manner as above. Of course one can also go back to the upper level. This procedure is repeated until the document (leaf) level is reached, and hopefully the user has found something interesting to look at.

Traditional search engines provide users with a way of (hopefully) locating interesting documents related to a query, but this requires the user to know how to express his needs using certain keywords to search for. Usually conventional retrieval systems return long lists of ranked documents that users are forced to scan through to find the relevant documents. On the web the high recall and low precision of the search engines makes this problem even worse. More over, the typical user has trouble formulating highly specified queries and does not take advantage of advanced search options. As if this was not enough, the problem gets worse as the Web or intranet grows. Our intention here is to present a system that provides another point of view; using an iterative procedure in a few steps, relevant documents can be found, even if the user did not know how to specify his query. In addition, the user may find potentially useful documents he did not know he was looking for! Such documents would probably not have been found using a traditional search engine, because the user probably would not have come up with a search string that would have generated those documents.

1.1 Background

The idea for this work began with a discussion with Henrik Fagrell on a spring day in 1997. He was doing some research in cooperation with Volvo, and we were looking for a subject for our thesis project. We had a few more discussions, and were introduced to Dick Stenmark at Volvo. He was at the time working with technical issues regarding Volvo’s intranet. Together we agreed on an assignment where we would be trying out techniques for clustering all available documents on Volvo’s intranet, in order to get an overview of it all.

1.2 Disposition

In the next chapter we will discuss the problem at stake of this thesis. We will also present how we will break it down in smaller subsections. The overall methods that we have been using throughout the process will be discussed in chapter 3. Chapter 4 gives an introduction to concepts and theories that provide a framework for understanding the problem and related issues. Next, we will in more detail present the techniques and algorithms that we have based our prototype upon, typically related to Information Retrieval. This is done in chapter 5. In the following chapter, we will review and discuss our actual implementation, step by step, rounded up with an informal evaluation. Chapter 7 provides further discussions of our system, along with ideas for future work and conclusions. The next chapter provides some additional related information in the form of appendices. These appendices describe some of the mathematics in more detail, as well as give a brief look at what other researchers addressing similar problems have come up with. Finally there is a list of references.

1.3 Acknowledgements

We would like to express our gratitude to our supervisors.

Dick Stenmark, at Volvo Information Technology, has helped us a lot with contacts at Volvo, getting things settled, and has provided us with valuable feedback along the way. Additionally, he has had the most remarkable patience and tolerance with us. Who knows how many deadlines we have passed without him losing his temper?

Henrik Fagrell, at the Department of Informatics, was the one who first inspired us to take on this assignment, and has guided and assisted us in numerous ways.

2 Problem description

In today's fast evolving and often geographically distributed companies, it is of great importance to be able to access crucial information in a fast manner. Many organizations are depending on fast decisions and effective information management to be able to keep up with the ever-hardening competition. This is not an easy task when their sources of information, much on behalf of emerging intranets, are growing almost in an exponential speed. But not just companies are facing this dilemma, it applies to almost everyone in our modern society: individuals and all sorts of organizations.

2.1 Problem definition

Here we present the problem definition, or research question, that we are addressing in this thesis:

> *How can the organization and structure of large, dynamic, networked and possibly very diverse text-based information collections be visualized, using clustering techniques?*

There are several reasons for this definition. We are explicitly studying a case at a large intranet, but will try to make some generalizations of the results, so we need a broader definition that could be applied to other types of networked information systems than intranets. But still, we cannot look at every possible way to address the information overload problem, so we focus on some specific techniques that could be used for this purpose. In particular, we are limiting our work to text-based information, i.e. documents, and we have focused on clustering techniques, since this seemed to be a promising way of handling the vast amounts of information we implicitly had to deal with.

2.2 Problem decomposition

In our early discussions with Volvo, we decided to try to apply some sort of clustering techniques to the documents available from their intranet. However, in order to do this, there are a number of prerequisites that need to be addressed first. Text documents, in our case mostly HTML files, can not be clustered the way they are, since there are no apparent means for automatic (i.e. suitable for a machine) comparison of the documents content. We have to carefully examine our options and prior research in the area to reach a satisfactory solution.

The problem can be divided into three major components that have to be considered (Oard and Marchionini 1996):

- Collection

At Volvo, there already is an internal search engine, and we are able to take advantage of the collection component of this system. There is already a working intranet robot (see chapter 4), that gathers all documents it can find on the intranet, and we are able to use this as raw input to our own system. This means that the collection part of our system is already taken care of.

- Selection

The main problem we are addressing concerns selection. This activity can be subdivided in many ways. As wee se it, there are three major decisions we have to make. First, we must realize some sort of *representation* of the documents that allows for clustering. Second, we need to know how to compare these representations with each other, in other words, we need to define our *measures of association* within the representation space. Third, we must decide on which *clustering scheme* to apply to these representations in order to organize them. These issues will be discussed in more detail in chapter 4.

- Display

The document clustering produces a hierarchical tree structure, which describes the relationships of the documents to each other and in a larger sense, the entire document collection, i.e. intranet in our case. We have developed a tool that allows for interactively browsing of this tree structure.

3 Method

In this chapter we will review the methodology we have used during our work, and discuss questions such as how and why we chose to go in a certain direction.

3.1 Literature review

It has been documented to be very effective to combine different approaches and disciplines when conducting informatics research. That is, even though a particular perspective is adopted as the main focus, research efforts are very likely to be more successful when combined and confronted with other, related disciplines. The research field related to Information Retrieval is traditionally a multidisciplinary approach, which is trying to combine research traditions from areas such as library and information science, linguistics, mathematics, social science, and computer science. This approach seemed to be a reasonable way of addressing our problem.

Bearing this in mind, we went forward with our literature survey. We wanted to get a good picture of previous work in the area, and not just stay tight to one perspective that might be dominant in a single discipline.

3.1.1 Literature gathering

We have conducted a thoroughly search of previous work on related subjects. Some of this information has been collected from books and journals, but the major part originates from various sites on the Internet. Early on in our work we started with trying to accomplish a detailed study of the sources on the Internet, and this path has been followed since then.

To get a feeling of were to begin our search on the Internet we started with the annually *International World Wide Web Conferences*, especially the latter years. There we found many interesting papers, of course not concerning our subject in every way, but with lots of comprehensive references to give us an idea of were to begin. Reading these papers gave us ideas of how to continue, and what type of information to look for.

Inspired by these articles, we could go on and search the entire (well, that parts that have been indexed by the major search engines) Internet using expressions like, *Unsupervised machine learning, Automatic Text Clustering, Information Retrieval, Information Filtering, Artificial Neural Networks, Classification, Categorization, Vector-Space model* and a couple of others. These expressions were used as queries to common search engines like AltaVista, Excite, HotBot, InfoSeek, MetaCrawler, and others. However, being overwhelmed with thousands of documents returned by the search engines, we realized that we had to find alternative strategies to find the information we wanted.

One strategy that usually provided high quality answers was to ask a human expert on the subject. However, it was hard to find someone in Göteborg that had exactly the expertise we were looking for, but still, we received much help by asking people at the Department of Informatics and Volvo for advice on what we should read.

Another strategy was trying to find homepages on the web that were related to our problem. These pages were usually maintained by either an individual researcher or a research group, and they proved to be a really valuable source. In addition to collections of published articles, easily available for downloading and printing, these sites had high quality links to similar sites that provided even more of what we were looking for. One way of originally finding such homepages was to use a search engine and make a subject search as described above. Another way was to follow up on references in articles that we had already read and knew were interesting. Since the names of the authors were there, usually accompanied by the article titles and work place, we could use these as input to the search engines and find the authors' homepages, and hopefully the original articles that we were looking for. Much more efficient than going to the local library. Other types of web sites could be searched for and located in similar ways. We found the homepages of e.g. many conferences, workshops and foreign university departments, which all in some way helped to solve the puzzle.

In addition the World Wide Web, we searched the different University Libraries in Sweden for books and authors that we came across during the inventory of the Internet and at various conferences. However, this strategy was not as effective as searching the Web, so we did not use it that much.

What we could not find on the Internet is of course information about Volvo's intranet, due to the security firewalls it is surrounded by. To do this we had to go and visit the company at Torslanda, Göteborg.

3.1.2 Literature analysis

Some of the theories and methods we have found useful during the literature study are described further down in the text. With useful information, we mean information that could help us to solve the problem at hand. However, it was quite difficult to judge the different papers and theories that we found in a proper way. The authors of the articles are experienced scientists who have put many years on a subject, and therefore much of the discussions are quite advanced. Also the algorithms used are anything but trivial in most cases.

We became quite aware of a problem that we ourselves were addressing, namely information overload. When we first set out to find relevant literature and learn more about the problem domain, we thought there would not be so much written about the subject of e.g. document clustering, and thus, it would be hard to find articles or books concerning this matter. But we were wrong! Even a seemingly narrow problem description such as 'document clustering' proved to have been the subject of research for some fifty years until now and literally shelf meters have been written on the subject. We were overwhelmed with articles and books concerning this matter in some way, and it became increasingly difficult to select the most valuable articles in this stream. We were indeed victims of information overload. How ironic, since this was one of the problems we were trying to resolve.

3.2 Search for suitable software

When we realized that so much research had already been carried out on related subjects, we started to look for implementations and solutions to various related problems. Another thing that came to mind was that the problem we were addressing was far more complicated than we thought at first. If we were going to actually implement something that would take advantage of previous research, we just could not start from scratch with our own system implementation. Another conclusion was that if we did not take previous research efforts into account, we probably would not be able to produce anything of value. So we decided on trying to find usable system components, that would allow us to build the system we wanted. The search for this software was done in very much the same way as for papers and articles. If a piece of software was found on a research site, it was generally accompanied by a couple of academic papers describing it. Quite often the software came with a user guide and examples of how to use it, even with concrete examples, but this was not always the case.

We also had to consider legal issues. Most researchers who posted software on their homepages would let anyone use it without restrictions. Some other provided it for free for academic purposes only. In one case we had to print, sign and mail a physical paper with a non-disclosure agreement in order to obtain a package of research software.

However, obtaining software was not a major problem, but evaluating it was problematic. Almost everything we found came in source code, with include files and options specified for some computer system that was different from ours. We spent many hours just figuring out how to compile these programs into executables.

When this was done, we had to test the software with our own data, which sometimes meant having to rewrite parts of the source code, or at least a lot of tinkering with parameters. Since some of the programs we tested were intended for completely different applications, e.g. graphical image clustering, this was not always an easy task. But then again, some of the programs we found were straightforward to use and worked reasonably at the first try.

At the point of these tests, we had not yet decided in detail on how we wanted our system to behave, so partially this software search was in blindfold, trying to find something that would suit our needs. Concurrently with the software tests we were reading articles on related theory. Slowly the picture of what we wanted to accomplish began to clear, and we eventually found some tools that would help us get there. Still, there was much work left to get everything to work together.

3.3 Prototyping

The method we have used for the system development could be characterized as a variant of “evolutionary prototyping” (Sommerville 1996). This is based on the idea of developing an initial implementation and exposing it to user comment and refining this through many stages until the system satisfies a potential user or user group. However, since we are not aiming towards an end-user product but rather we want to test some ideas and how they may work out, we have used ourselves and local expertise, instead of a user community in the manner that Sommerville suggests.

Sommerville argues that this is the only realistic way to develop systems where it is difficult or unrealistic to make a detailed system specification, and this surely complies with the system we are tying to build. He also means that this approach is very suitable for the development of systems that attempts to emulate some human capabilities like our system does. To be able to succeed in this approach, Sommerville argues for the use techniques that makes it possible to do rapid systems iterations, whereas one can quickly evaluate changes in the system and immediately make corrections or include new features. Another point of view is the difference between traditional specification-based approach and evolutionary prototyping is the viewing of verification and validation. As Sommerville put it, verification is only necessary when there is a specification to compare it with. If there is not a specification there is not very much to do the verification against. On the other hand, the validation process should show that the system or program is suitable for the intended purpose rather than a perfect conformance to a predefined specification.

We have applied this evolutionary method to our setting, and produced an initial prototype, which as been further refined in many, many iterations. However, the objective we had in mind mainly concerned testing some ideas and developing a prototype, and not reaching a level where we would produce detailed specifications.

4 Background

In this chapter we will present and discuss some theories and ideas that we have found relevant to understanding the problem domain, as well as examining Volvo's intranet, where we have carried out our work.

4.1 Information overload

With the growth of the Internet and other networked information, there is no problem *finding* information[1], instead the problem is to find the *right* information. Why is that? The amount of information available is growing at an almost exponential rate and our possibilities to get a hold of it are diminishing, as Wurman (1989) wrote in his book Information Anxiety. This phenomenon is usually recognized as the *information overload* problem. This problem was already observed and discussed by Vannevar Bush (1945).

Information overload could be seen as the diagnosis for an individual being presented an amount of information exceeding his or her cognitive capacity. A similar concept, information anxiety, is the primary defining characteristic or result (symptom) of the information overload problem. If a person did not have any problems finding the correct information, or if the information came in just the right quantity, then information overload would not exist. Information anxiety results from our inability to access and extract meaning from the wide accumulation of information available to us (Nelson 1994).

However, Ljungberg and Sørensen (1998) presents some critical viewpoints to this definition, as he points out that information overload is a concept stemming from a database oriented view of information technology. It focuses on situations where the amount of information exceeds the cognitive capacity of the recipient of the information. It does not focus on communication patterns, and information overload is often exemplified by the difficulties related to information retrieval in large databases. In order to reduce the risk of facing information overload, the amount of information must be reduced, either by inventing more effective tools for information processing, e.g., information retrieval or filtering, or by increasing our cognitive capacity, thereby processing the information more efficiently.

4.2 Conceptual framework of information seeking

Oard and Marchionini (1996) presented a conceptual framework which deals primarily with a related problem, namely Information Filtering, also known as selective dissemination of information in the Library and Information Sciences. This subject deals with sorting through large volumes of dynamically generated information, often seen as an information stream, and presenting the user with results, which are likely to satisfy his or her information requirement, using some sort of filtering, i.e. selecting what should pass according to a relatively stable profile.

We will try to adopt parts of this framework that are applicable to our problem domain. Oard and Marchionini use the term "information seeking" as an overarching term to describe any processes

[1] It is common to draw a distinction between information and data in which the concept of "information" includes some basis for its interpretation. In this work, however, we combine the two concepts and refer to both as "information".

by which users seek to obtain information from automated information systems. The overall goal is to present users with direct information, or information sources that are likely to satisfy his or her information requirement. “Information sources” refer to entities, which contain information in a form that can be interpreted by a user. Information sources, which contain text, are commonly referred to as “documents”, but in other contexts these sources may be audio, still or moving images, or even people. In this work we will focus on text-only based information sources.

Collection
Selection
Display

Figure 1. Information seeking task (Oard and Marchionini 1996)

The process of information seeking can be divided into three subtasks: *collecting* the information sources, *selecting* the information sources, and *displaying* the information sources, as shown in figure 1.

The distinction between process and system is fundamental to understanding the difference between different information seeking activities, e.g. information filtering and information retrieval. By “process” we mean an activity conducted by humans, perhaps with the assistance of a machine. When we refer to a type of “system” we mean an automated system (i.e. a machine)

4.3 Navigation aids

When dealing with Internet technology, particularly the World Wide Web, or and intranet the uses the same technology basis, access to the information is commonly performed through browsing in some way. Since the hypertext structure allows online documents to be connected in almost any way, there is an incredible number of ways one could use when navigating across pages, looking for suitable information. Clearly, it is not always possible to find what one is looking for through browsing only.

This problem has been recognized long ago, and much research and effort has been put in trying making it easier to find what one is looking for. There are many propositions that addresses this problem in the literature, and many of them have found their way into real life applications, such as various search engines and agent approaches. We will, however, focus on techniques that are part of the installed base at Volvo in this discussion.

One common example is to have link collections, which provides a good overview of available resources. Many people provide personal link pages that applies to their interests, and some attempts have been made to categorize a substantial part of the overall available Web resources, e.g. by Yahoo! Inc. However, building these category structures, organized around topics in a hierarchical manner require a considerable amount of human labor efforts, and it is virtually impossible to keep up with the dynamic changes on the Web.

Another approach is taken by the search engines, e.g. AltaVista, HotBot and Excite. These services rely on automatic indexing techniques, and ‘robots’ that automatically and repeatedly scan the Web for new documents. Still, it can really be hard to find what you want, when you get some 100,000 hits as result from a search query.

Both these approaches are present on Volvo’s intranet. From a user’s point of view, they are somewhat complimentary. If you know what you are looking for, and are able to express that information need in a search query, you could hopefully get some good results from the search engine. Since the search engine relies on automatic and continuos gathering and indexing of documents, it is supposed to be reasonably good at keeping up with the evolving structure of the intranet. This is not necessary the case with the Yahoo-style IntraPages, that are manually constructed. However, this interface has the advantage that could present an overview of everything that is present, and as a user, you are able to browse through the organized (usually in a hierarchical fashion) structure, just to see what is out there, and get some inspiration. This could be helpful if you cannot express your information needs in the formal manner that is required by the search engine’s interface. When browsing an organized index in this way, you might stumble over something that could turn out really valuable to you, and that you would never have thought of making a query for. In this way, these two approaches to support Web, or in this case, intranet navigation somewhat compliment each other, but still both approaches are dealing with serious drawbacks.

4.4 Intranets

In our empirical work, we have conducted hands-on work with Volvo’s intranet. In this section we will try to give a brief description of what an intranet is, and what it is used for.

An intranet is a private corporate network based on internet's protocols and technologies. At the foundation of the intranet are one or several Web servers, which are used to manage and disseminate information within the organization (Lai and Mahapatra 1997). Using a standard Web browser as an interface, employees can exchange corporate information seamlessly without the concern of heterogeneous computing environment. With organizations under immense pressure to empower employees and to better leverage internal information resources, intranets can serve as a highly effective communications platform to disseminate information for the entire organization, including its remote offices.

Increasingly, proactive corporations are taking advantage of intranets to disseminate company documents, forms, news, policies, phone directories, product specifications, and pricing information. A survey from 1996 conducted on Fortune 1000 companies indicated that twenty-two percent of them were already using Web servers for internal applications; while another forty percent were considering the implementation of intranets to make their information more readily available. In order to reap the full benefits of intranets, organizations are extending their intranets to reach their key customers, suppliers, and/or trading partners. They also support team-oriented collaboration, including file sharing, information exchange, document publishing, and group discussion.

In addition to using intranets to integrate individual, group, departmental, and corporate communications, business managers in a number of industries are beginning to identify strategic opportunities for using intranets to shift the balance of power and competitive position of their organization. Some are thinking of adopting intranets as a tool to unify their geographically dispersed work force, empowering them (especially telecommuters and sales forces on the road) with a complete communication tool for collaboration, interaction, and real-time sharing of information across functional boundaries and organizational levels. This new form of distributed information infrastructure may even enable corporate managers to redefine their computing strategy and organizational control to better accommodate the challenges of managing speed and complexity in today's business environment.

4.5 Volvo's intranet

The intranet of Volvo has since it was introduced 1995 grown from nothing to approximately 100 servers. As PCs are getting more powerful, servers will continue to grow in numbers. Like the Internet itself, the intranet is highly decentralized – any one can download and operate a web server, and they will! It is already impossible to enforce a standardized view or a central list of the resources. The number of users is exceeding and it is easy to publish almost anything you like. But can you be sure that anyone is going to find it? Today Volvo Information Technology has a rudimentary search tool called VISIT[1], which in turn is based on Harvest[2] - a search tool developed at the University of Colorado at Bolder. The application is an integrated set of tools to gather, extract, organize, search, cache and replicate relevant information across Internet, or as in this case, an intranet (Hardy, Schwartz et al. 1996). The Harvest search tool produces an index, containing pointers to all documents available on the intranet and other information, such as author, production time, content et cetera.

[1] *Volvo Intranet Search Indexing Tool*
[2] *http://harvest.transarc.com*

5 Techniques and algorithms

In this chapter we will present an overview of the models and theories that we have based our prototype upon, together with related concepts. Most of this work originates from the research field of Information Retrieval (IR), which concerns the problem of retrieving those documents from a given document-base that are *likely* to be *relevant* to a certain information need. In the following, we will mainly discuss Information Retrieval in the meaning of text or document retrieval, and disregard other types of media, such as sound, video, speech, and images.

Some of the theories described in this chapter have evolved from other disciplines, such as linguistics and mathematics, but have important applications in IR.

5.1 Automatic Text Analysis

Before a computerized information retrieval system actually can operate to retrieve the information that a user has searched for, that information must, of course, already have been stored somewhere.

A starting point of a text analysis process may be the complete document corpus, an abstract, the title only or perhaps a list of words only. The frequency of a word occurrence in a document provides a useful measurement of word significance. It is proposed that the relative position of a word within a sentence, having given values of significance and provides a useful measurement for determining the significance of sentences. The significance factor of a sentence will therefore be based on a combination of these two measurements (Luhn 1957). The idea is that frequency data can be used to extract words and sentences to represent a document.

If we let *f* represent the frequency of occurrences of various word types in a given position of text and *r* their rank order (the order of their frequency of occurrence), then a plot relating *f* and *r* yields a curve similar to the one hyperbolic curve in *figure 1.* This one demonstrates Zipf’s Law, which states that the product of the frequency of use of words and the rank order is approximately constant.

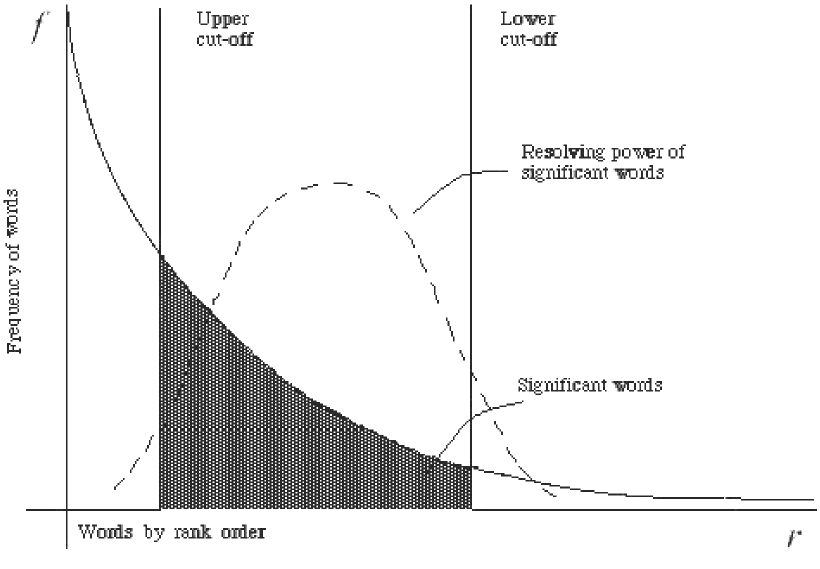


Figure 1. A curve plot relating the frequency f and the rank order r.

The curve is used to specify two cut-offs, an upper and lower which excludes non-significant words. The words exceeding the upper cut-off were considered to be common and those below the lower cut-off rare, and therefore not contributing to significantly to the content of a specific document. The resolving power of significant words, i.e. words that discriminate the specific content of the actual text, reached a peak at a rank order position half way between the two cut-offs and from the peak fell off in either direction reducing to almost zero at the cut of points. The cut off points is determined by applying trial and error, there is no given values.

5.2 Information Retrieval from the Web

Work in information retrieval systems goes back many years and is well developed (van Rijsbergen 1979; Salton 1989; Belkin and Croft 1992). However, most of the research on information retrieval systems is on small, well-controlled and relatively homogeneous collections such as collections of scientific papers or news stories on a related topic. Indeed, the primary benchmark for information retrieval, the Text Retrieval Conference, uses a fairly small, well-controlled collection for their benchmarks. Things that work well on TREC often do not produce good results on the web. For example, the standard vector space model tries to return the document that most closely approximates the query, given that both query and document are vectors defined by their word occurrence. On the web, this strategy often returns very short documents that are the query plus a few words.

With the advent of large distributed and dynamic document collections (such as are on the World Wide Web), it is becoming increasingly important to automate the task of text categorization (Liere and Tadepalli 1996). For example, the idea of document clustering, i.e. automatic organization of documents according to some criteria, e.g. semantic similarity, has been on the research agenda for many years (van Rijsbergen 1979; Salton 1989).

5.2.1 Libraries versus the Web

Most of the previous research in the Information Retrieval field aimed at static or semi-static document collections, related to libraries and long texts. Our research context, a web system, differs from previous ones in that it is highly dynamic, and many documents are fairly short in length. The organization of data is very different from conventional libraries. This applies to “normal” digital libraries as well, which are essentially digitized versions of the former. They consist of relatively long text documents that are well organized, by means of human effort. Web or intranet documents typically have a much less degree of organization, and are physically distributed and scattered in a way that has no correspondence in libraries. The web is a vast collection of completely uncontrolled heterogeneous documents. Documents on the web have extreme variation internal to the documents, and also in the external meta information that might be available. For example, documents differ internally in their language (both human and programming), vocabulary (email addresses, links, zip codes, phone numbers, product numbers), type or format (text, HTML, PDF, images, sounds), and may even be machine generated (log files or output from a database). Another big difference between the web and traditional well controlled collections is that there is virtually no control over what people can put on the web.

This indicates that we may have a log to learn from the long tradition within Information Retrieval, but we have to face the new problems that arise when applying these techniques to web-based systems such as an intranet (Brin and Page 1998).

5.2.2 Web robots

When collecting data from the Web-based networks, be it local intranets or the World Wide Web itself, it is common to use so-called web robots, which also are referred to as 'Web Wanderers', 'Web Crawlers', or 'Spiders' (Brin and Page 1998). These names are, however, misleading as they give the impression that the software itself moves between sites like a virus. This is not the case, a web robot simply visits sites by requesting documents from them. Web robots are also used by many Web search engines (e.g. AltaVista, Lycos) to collect data for indexing. A (web) robot is basically a program that automatically traverses the Web's hypertext structure by retrieving a document, and recursively retrieving all documents referenced. The term 'recursively' does not limit the definition to any specific traversal algorithm. The robot can apply some heuristic algorithm to the selection and order of documents to visit, it is still just a robot. A web browser is not in itself a robot since it is operated by a human user and does not automatically retrieve referenced documents. If the robot does not contain rules stipulating when to stop, it might attempt to retrieve all the public pages on the Web. The criteria for stopping can be defined relative to a certain depth in the link structure, or when a predefined number of documents have been retrieved. There is a common agreement in the web robots community concerning certain ethical rules (Eichmann 1994) (Koster 1995) that robots have to follow. These rules regard issues such as avoiding to squire resources from human users by retrieving pages at high speed. The robot must also identify itself to the web server so that the webmaster can contact the owner of the robot if problems occur. An example of such a problem might be when the robot is getting stuck in a 'black hole', which is a page with a script designed to generate a new page when accessed. This detains the robot until its owner shuts it down, possible after it has caused nasty network delays or filled a disk with useless data.

5.3 Definitions of Information Retrieval

Here are two attempts to define Information Retrieval.

Salton (1989):

“Information-retrieval systems process files of records and requests for information, and identify and retrieve from the files certain records in response to the information requests. The retrieval of particular records depends on the similarity between the records and the queries, which in turn is measured by comparing the values of certain attributes to records and information requests.”

Kowalski (1997):

“An Information Retrieval System is a system that is capable of storage, retrieval, and maintenance of information. Information in this context can be composed of text (including numeric and date data), images, audio, video, and other multi-media objects.”

Some years have passed between these two definitions, and the development of distributed networked information systems in general, and perhaps the Internet in particular, seems to have affected the latter of the two. Salton’s definition has something of a database metaphor over it, while Kovaliski's definition is broader and applies much better to the kind of systems we are dealing with.

5.4 Traditional Information Retrieval

Information retrieval has since the 1940's been attracting increasing attention. As we already have mentioned, there is a vast amount of information, to which fast and accurate retrieval is becoming more and more difficult to accomplish. One consequence could be that relevant information never is discovered because it is almost impossible to find it. Thanks to the advent of computers many problems with storing large amounts of data has in the last decades been solved. Never the less, there still is much more to do to make information retrieval effective (van Rijsbergen 1979).

The main purpose with information retrieval is relevance. This is because that what's its all about – to retrieve all the relevant documents and at the same time retrieve as few non-relevant documents as possible. The process of information retrieval can be illustrated with a black box system as in the figure below. The users' query and the existing documents is the input to the system. When the user get an output result, he or she may apply feedback to the system in order to change the query to get a better result in the next search. However, this feedback component is not present in all information retrieval systems.

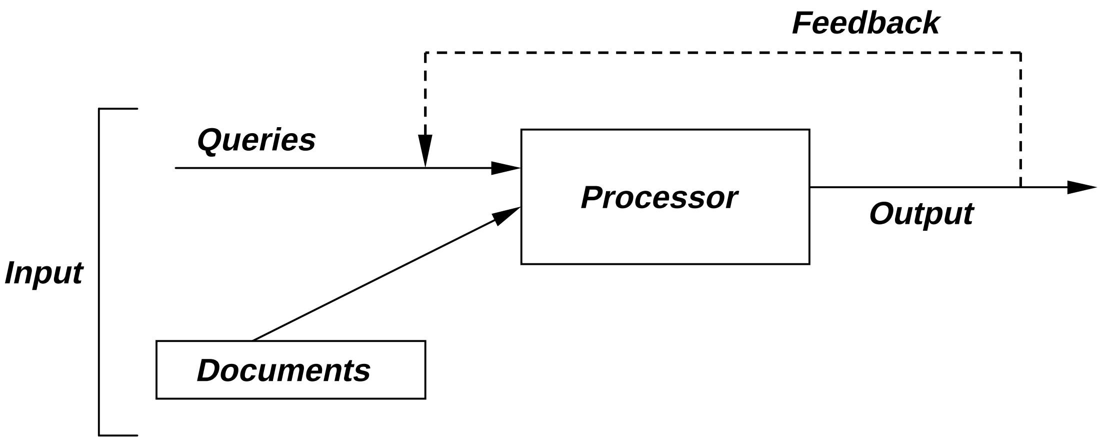


Figure 1. A typical Information Retrieval system (van Rijsbergen 1979)

We have found that much of the research and development in information retrieval is aimed at improving the effectiveness and efficiency of retrieval. *Efficiency*[1] is usually measured in terms of the computer resources used such as core, backing store, and CPU time (van Rijsbergen 1979). There is a difficulty in measuring effectiveness in a machine independent way. It should be measured in conjunction with effectiveness to be able to obtain some idea of benefits in terms of unit cost.

Effectiveness[2] is commonly measured in terms of precision and recall where precision is the ratio of the number of relevant documents retrieved to the total number of documents received. Recall in turn, is the number of relevant documents retrieved to the total number of relevant documents,

[1] *Efficiency is to do the thing right*

[2] *Effectiveness is doing the right thing*

both retrieved and not retrieved. It has been shown (Lesk & Salton, 1969) that a subsequently scale on which a document is either relevant or non-relevant, when subjected to a certain probability of error, did not invalidate the results obtained for evaluation in terms of precision and recall.

5.4.1 Information Retrieval Models

Most existing methods of text categorization and text retrieval fall into one of three categories: Boolean, probabilistic or vector space (Belkin & Croft, 1992).

Boolean is based on the concept of exact match of a search string expression or phrase. Here all texts containing the search string specified in the query, are retrieved. One drawback is that there is no distinction made between the retrieved documents. ***Probabilistic*** information retrieval models are based on the probability ranking principle, which states that the function of information retrieval systems is to rank the text in a database. This would make up an order of their probability of relevance to the query. Finally the ***vector-space model***, which treats texts and queries as vectors in a multidimensional space, and the dimensions are words, which is used to represent the documents. The search strings are compared by comparing the vectors and the of use for example cosine correlation similarity measure. The assumption is that the more similar a vector that is representing a text is to a query vector, the more likely that the text is relevant to the query. We will describe this different approaches further on in this thesis.

Many older IR systems are based on inverted indices, which, for each keyword in the language, store a list of documents containing that keyword.

Various enhancements have been proposed to improve the accuracy of inverted index queries. Most of these enhancements are “labor intensive”; that is, they ultimately require the user to be more specific. One such improvement is the ability to create sets of documents corresponding to an individual keyword and then to manipulate those sets using Boolean logic. AND, OR, NOT etc.

5.4.2 The Origin of Vector Space Models

In 1953, H.P. Luhn published an initial discussion of vector-space models for information retrieval that summarized many of the key issues and concepts still being considered today. Luhn was motivated by the concern that the controlled vocabularies and classification schemes used in manual indexing may change over time. Luhn was also concerned that by only classifying concepts in a document that seemed important at the time, aspects of the document that might become more important in the future would be lost.

5.4.3 The Vector Space Model

The Vector Space Model of Information retrieval provides an alternative to the Boolean model, which allows more accurate automatic document classification.

Instead of storing a list of documents and frequencies for each keyword, as in the Inverted Index, we store a list of keywords and their frequency for each document. Thus every document becomes a vector in n dimensional space where n is the number of keywords in the language. The Vector Space Model is based on the assumption that similar documents will be represented by similar vectors in the n-dimensional vector space. In particular, similar documents are expected to have small angles between their corresponding vectors.

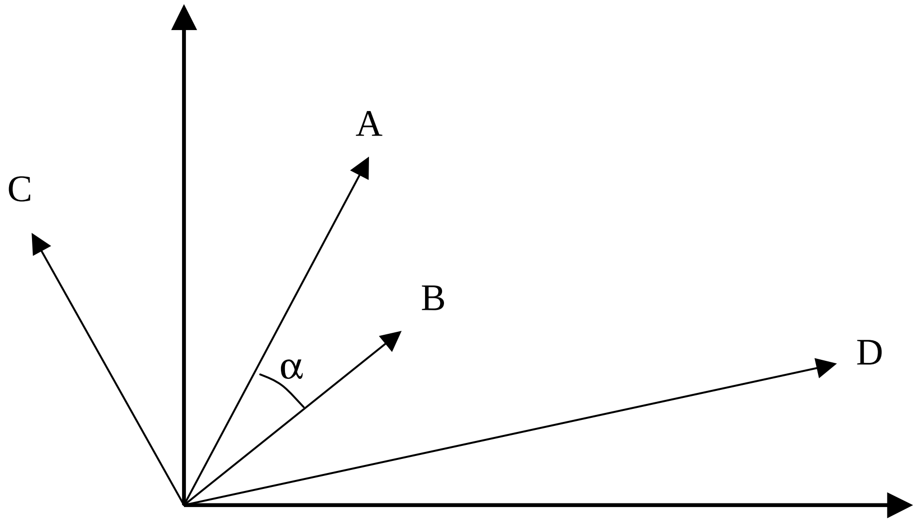


Figure 1: A simplified view of a two-dimensional vector space.

The vector-space models for information retrieval are just one subclass of retrieval techniques that have been studied in recent years. The taxonomy provided in [BC87] labels the class of techniques that resemble vector-space models "formal, feature-based, individual, partial match" retrieval techniques since they typically rely on an underlying, formal mathematical model for retrieval. These techniques model the documents as sets of terms that can be individually weighted and manipulated, perform queries by comparing the representation of the query to the representation of each document in the space, and can retrieve documents that don't necessarily contain one of the search terms. Although the vector-space techniques share common characteristics with other techniques in the information retrieval hierarchy, they all share a core set of similarities that justify their own class.

Vector-space models rely on the premise that the meaning of a document can be derived from the document's constituent terms. They represent documents as vectors of frequencies of terms, where each unique term in the document collection corresponds to a dimension in the space. Similarly, a term or query is represented as a vector in the same linear space.

The document vectors and the query vector provide the locations of the objects in the term-document space. By computing the similarity between the query and other objects in the space, objects with similar semantic content to the query presumably will be retrieved.

Vector-space models that don't attempt to reduce or collapse the dimensions of the space treat each term independently, essentially mimicking an inverted index (Frakes and Baeza-Yates 1992). However, vector-space models are more flexible than inverted indices since each term can be individually weighted, allowing that term to become more or less important within a document or the entire document collection as a whole. Also, by applying different similarity measures to compare queries to terms and documents, properties of the document collection can be emphasized or de-emphasized. For example, the dot product (or, inner product) similarity measure finds the Euclidean distance between the query and a term or document in the space. The cosine similarity measure, on the other hand, by computing the angle between the query and a term or document rather than the distance, de-emphasizes the lengths of the vectors. In some cases, the directions of the vectors are a more reliable indication of the semantic similarities of the objects than the distance between the objects in the term-document space (Frakes and Baeza-Yates 1992).

Vector-space models were developed trying to overcome many of the problems associated with exact, lexical matching techniques. In particular, since words often have multiple-meanings (polysemy), it is difficult for a lexical matching technique to differentiate between two documents that share a given word, but use it differently, without understanding the context in which the word was used. Also, since there are many ways to describe a given concept (synonymy), related documents may not use the same terminology to describe their shared concepts. A query using the terminology of one document will not retrieve the other related documents. In the worst case, a query using terminology different than that used by related documents in the collection may not retrieve any documents using lexical matching, even though the collection contains related documents (Berry, Dumais et al. 1995). Vector-space models, by placing terms, documents, and queries in a term-document space and computing similarities between the queries and the terms or documents, allow the results of a query to be ranked according to the similarity measure used. Unlike lexical matching techniques that provide no ranking or a very crude ranking scheme (for example, ranking one document before another document because it contains more occurrences of the search terms), the vector-space models, by basing their rankings on the Euclidean distance or the angle measure between the query and terms or documents in the space, are able to automatically guide the user to documents that might be more conceptually similar and of greater use than other documents.

Also, by representing terms and documents in the same space, vector-space models often provide an elegant method of implementing relevance feedback [SB90]. Relevance feedback, by allowing documents as well as terms to form the query, and using the terms in those documents to supplement the query, increases the length and precision of the query, helping the user to more accurately specify what he or she desires from the search.

5.5 Term weighting

Experience has shown that information retrieval effectiveness can be significantly improved by transforming the raw term-frequency vector in ways which amplify the influence of words, which occur often in a document, but relative rarely in the whole collection of documents. One common scheme is “term-frequency – inverse document frequency” (tf-idf) weighting. This scheme assigns term *i* in document k *a* weight value computed as:

The benefits of term weighting are well-known in the information retrieval community, beginning with the work by Luhn, Jones & Kay, and others (van Rijsbergen 1979). Terms occurring frequently in a collection, while skewing the results of queries that include those terms, are essential for describing the relationships between the documents. However, to better differentiate between documents, non-frequently occurring terms in a collection should be given greater weighting value than frequently occurring terms when determining the similarity between documents. Weighting can be either global, applied to and determined from the entire document collection, or local, with effect only on a single document. Empirical studies suggest that combining these methods achieve the best performance issues (van Rijsbergen 1979; Salton 1989). Both global and local weighting is implemented as a mathematical function which returns a value in the interval [0,1]. The final weighting, applied to each non-zero element in the term by document matrix, is the product between the global and local weighting. (Dumais 1991) found that combined with LSI, the log-entropy weighting scheme provided a 40% advantage over raw term frequency on several standard document test collections.

(Bartell, Cottrell et al. 1992), using a mathematical approach, concluded that certain term weighting in the original term space, in combination with LSI, should improve the performance of the technique.

5.5.1 Global Weighting

In 1972, Sparck Jones [Jon72] examined the use of global weighting schemes to improve the performance of information retrieval systems. The author argued that terms occurring frequently in a collection, while skewing the results of queries that include those terms, were essential for effective information retrieval. However, to better differentiate between documents, non-frequently occurring terms in a collection should be given greater value than frequently occurring terms when determining which documents in a collection were relevant to a query. On each of the three document collections used in a test, the performance improvements provided by global weighting surpassed that of any other single improvement to information retrieval systems suggested in the literature. Together with local weighting, the global weightings substantially increased the retrieval performance of even simple retrieval systems.

5.6 Feature selection and Dimensionality reduction

5.6.1 Stemming

Stemming means reducing different word forms to common roots. The purpose of stemming is to group words that are morphological variants on the same word stem. This technique is also known as suffix stripping or term truncation (van Rijsbergen 1979; Salton 1989). In the English language, stemming is traditionally carried out by stripping off the common suffixes, and the Porter and the Lovins stemmer are two widely used implementations. To perform stemming in Swedish, somewhat more complicated algorithms must be carried out, based on similarity between sounds as they are spoken, translated into phoneme (a few letters indicating a sound), among others. The major problem with this type of morphological stemmers is that they make mistakes because they do not pay attention to the meanings of words, i.e. the semantics. There exist more advanced stemmers that also take into account semantic relationships between words obtained from a machine-readable dictionary, such as a semantic net. Although many mistakes made by morphological-only stemmers may be avoided, the effectiveness is not consistently better, because it is often too conservative. For example, the words “stock” and “stocks” might not stem together, because the words may have different meanings, While in some cases this separation is good, if definitively hurts when stemming is performed on a financial text corpus.

The value of stemming has been questioned in the past, but recent studies have shown that stemming can produce consistent though small improvements in retrieval effectiveness over a large range of collections. When dealing with relatively short documents, stemming has even higher value. Since many of the documents we are dealing with are very short, this indicates that the use of a stemming component could prove valuable.

5.6.2 Stop Words

(van Rijsbergen 1979) p.104

In written text, some words are very common and have no additional meaning to the actual content of the text, and has little or nothing to say about the text itself. Prepositions, conjunctions, nouns and articles are examples of such words. A lot of processing time and working memory can be saved if the words that does not contribute to the actual content of the corpus are removed. The percentage that the corpus is being decreased is often about 70 – 75 % compared with before the reduction was done. This is done by filtering the term list with so called “stop-lists” or “fluff-word lists”, see table 3.2, consisting of all words defined as not meaning-bearing, also called syncategorematic or non-context bearing words (van Rijsbergen 1979).

5.6.3 Reduced-Space Models

A short time after Luhn’s ideas were published, H. Borko and M. Bernick [BB63] presented a method by which documents could automatically be classified into predefined categories.

To define the classification categories, Borko and Bernick constructed a term-document matrix for their experimental document collection, using the frequencies of the terms in each document as the elements of the matrix. They then computed correlation coefficients for each term against all the other terms in the matrix. After the correlation matrix was subjected to factor analysis [Har47] to reduce its dimensionality, a set of orthogonal factors was extracted, rotated, and interpreted as classification categories.

Once the classification categories were determined, a prediction formula, based on term frequency and the normalized factor loading of terms in the given category, was devised to automatically classify the documents in the validation set. The prediction formula was applied to each document and category, and the category with the highest predicted result was selected as the most probable category for the document. Results indicated that approximately half of the documents were correctly classified by the system.

5.7 Measurements in IR systems

5.7.1 Measures of association

Discuss “aboutness” in IR...(Huibers 1996)

The similarity information can be metric (indicating the exact target similarities for the configuration of points), or non-metric (indicating only the relative ordering of inter-object similarities) (Bartell, Cottrell et al. 1992).

Similarity between vectors (representing documents, terms, or queries) can be measured in many ways. (Bartell, Cottrell et al. 1992) showed that LSI gives optimal similarity measures when using the inner (dot) product between two vectors as the similarity metric, though empirical studies have mostly used the cosine measure.

The similarity or association measure $M(\mathbf{X}, \mathbf{Y})$ between two n-dimensional vectors, also known as the proximity index, can be measured in many ways (van Rijsbergen 1979).

See appendix B for some mathematical examples of commonly used similarity measurements.

5.7.2 Evaluation performance of IR systems

Information retrieval models typically express the retrieval performance of the system in terms of two quantities: precision and recall. Precision is the ratio of the number of relevant documents retrieved by the system to the total number of documents retrieved. Recall is the ratio of the number of relevant documents retrieved for a query to the number of documents relevant to that query in the entire document collection. Both precision and recall are expressed as values between 0 and 1. An optimal retrieval system would provide precision and recall values of 1, although precision tends to decrease with greater recall in real-world systems [FBY92].

The performance of Information Retrieval systems is highly dependent on the content of a collection and is therefore difficult to evaluate objectively. To help evaluate the performance of these systems, the Information Retrieval community has developed two complementary measures: “recall” and “precision”. Recall is defined as the fraction of relevant documents in the data set which are returned as results to a given query. Precision is defined as the fraction of documents in the returned set which are actually relevant to the query. In mathematical terms:

Recall = relevant documents returned / all relevant documents

Precision = relevant documents returned / all returned documents

The relationship between recall and precision in information retrieval is analogous in many ways to the relationship between space and time found in certain other computer science models. Just as time requirements for certain algorithms cannot be improved beyond a point without sacrificing more memory space, improvements to either recall or precision are typically made at the expensive of the other.

Recall and precision are two well-known measures of the performance quality of an information retrieval system (Salton 1989). Defined as below, they give a good picture of how well an IR system performs and provides a means for comparing different approaches.

5.8 Latent Semantic Indexing

Latent Semantic Indexing (LSI) is a theory and method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text. The underlying idea is that the totality of information about all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and set of words to each other. In this way, LSI tries to overcome the problems of lexical matching by using statistically derived conceptual indices instead of individual words for retrieval (Berry, Dumais et al. 1995).

With LSI, we assume there is some underlying (latent) semantic structure in the data that is partially obscured by the randomness of word choice with respect to retrieval. Much of this "noise" can be eliminated by means of Singular Value Decomposition (SVD), a mathematical technique (Deerwester, Dumais et al. 1990).

Latent Semantic Indexing is a technique for representing documents, queries, and terms as vectors in a multidimensional, real-valued space. The objects are represented so that inter-point similarities match inter-object similarities between documents and terms. This is achieved using a powerful and fully automatic statistical algorithm, which can retrieve relevant documents even when they do not share any words with a query - concepts replace keywords to improve retrieval. LSI has been found to be an optimal special case of Multidimensional Scaling (MDS), a well-studied class of algorithms for analyzing inter-object similarity information (Bartell, Cottrell et al. 1992).

LSI is mainly intended for Information Retrieval, but it can be modified for many other tasks, e.g. term or document clustering, and modeling of the human learning process, known as Latent Semantic Analysis (LSA). Its learning mechanism is equivalent to a particular kind of linear neural network (Landauer, Laham et al. 1997).

Drawbacks:

The complexity of the LSI model often causes its execution efficiency to lag far behind the execution efficiency of the simpler, Boolean models, especially on large data sets (Letsche and Berry 1997). Most of the processing time needed for LSI is spent in computing the truncated document by term matrix. Retrieving answers to queries can be computed in a few seconds on a normal workstation, thus giving an acceptable delay for the user of the system.

Why not Boolean search?

Individual keywords are not adequate discriminators of semantic content. Rather, the indexing relationship between word and document content is many-to-many: A number of concepts can be indexed by a single term, and a number of terms can index a single concept. When retrieval is based solely on the matching of terms between the query and the documents, performance suffers as some relevant documents are missed (they are not indexed by the keywords used in the query, but by synonyms) and some irrelevant documents are retrieved (they are indexed by unintended senses of the keywords in the query, i.e. polysemy) (Furnas, Landauer et al. 1987) (Bartell, Cottrell et al. 1992).

5.8.1 Pros and cons with LSI

Compared to other concept-based approaches for Information Retrieval, LSI has several advantages (Deerwester, Dumais et al. 1990).

The LSI method has equaled or outperformed standard vector retrieval methods and other variants in almost every case, and was as much as 30% better in some cases (Dumais 1995) [WHY]. Since LSI uses a vector-space model rather than lexical comparison, documents and terms are both placed in the same semantic space, using the same kind of representation. This way, relevance feedback, or using full documents or clusters of documents rather than single terms is possible, and its just as simple as using only terms. Thus, one can search and find terms that are similar to a term, or find terms that characterize a particular document (Berry, Dumais et al. 1995).

LSI has its drawback as well. Location of terms in a document is not taken into consideration, which means that word order and sentence structure is completely ignored. A document regarding two or more separate topics will get a vector encoding in between these topics, and will not rank very for a search for either of these topics, and sometimes LSI fails to encode documents properly (Ladha 1998). If the encoding fails, the document may never be retrieved when a search for that topic is performed.

5.9 Clustering techniques

Given a large set of multi-dimensional data points, the data space is usually not uniformly occupied. Data clustering identifies the sparse and the crowded places, and hence discovers the overall distribution patterns of the dataset.

In cluster analysis, the idea is to assemble variables into unique groups or clusters of similar items. Several approaches to clustering may be encountered. The simplest one looks first for the two variables with the highest similarity to, or lowest distance from (measured in some way), any of the two members of this group, and so on. (Single-link, average-link). Cluster analyses can also be hierarchical or non-hierarchical. The hierarchical methods, which are the more common, link the individual clusters together so that each cluster is in turn a member of a higher level cluster, with the highest level-cluster incorporating all other clusters and items. The results of this type of clustering are typically presented as a tree diagram or dendrogram (agglomerative, conglomerative, = bottom-up, top-down). A non-hierarchical approach do not attempt to link the smaller clusters together, but they do on the other hand allow items to belong to more than one cluster (hard vs. soft (fuzzy)).

Document clustering can enhance retrieval effectiveness when the so-called cluster hypothesis holds: closely associated documents tend to be relevant to the same information request.

The organization of document clusters usually conforms to a tree-like structure (dendrogram). Smaller and tighter clusters are at lower levels and larger at coarser ones at higher levels. The clusters are composite in that each cluster consists of several smaller ones except for the leaf clusters that contain single documents. Each cluster has a representation, called its centroid. The centroid is a summary of the contents of the documents, which are its offspring in the tree. To generate the tree-like structure, the straightforward methods start with the similarity information between all pairs of documents in the entire collection. Usually the document pairs must be sorted by the similarity values. Hierarchical clustering can proceed either top-down or bottom-up. The bottom-up approach is also referred to as agglomerative clustering, because in each step it joins the two closest related clusters (or single documents) into a larger one.

There are three definitions of the closeness between two clusters: Single-link, complete-link and average-link. The single-link similarity between two clusters is the similarity between the two most similar documents, one of which appears in each cluster. The complete-link similarity is the similarity between the two most dissimilar documents, one from each cluster. The average-link similarity is a compromise between the two. With the same similarity value, a single-link cluster tends to be larger and looser than a complete-link one. There are also clustering methods, which do not require prior knowledge of the similarities between all pairs of documents.

Clustering identifies a finite set of categories to describe a set of data.

Traditional applications of clustering include discovering structure in data and providing summaries of data.

Clustering in the context of Information Retrieval mainly regards two approaches: document clustering and term clustering. The latter is tightly connected to thesauri building. A thesaurus can be used during a search session to increase recall by expanding search queries with related terms. In document clustering the search can retrieve documents similar to a specific document, even if the original query would not have retrieved that item. However, one has to bear in mind that the clustering process in not precise. It is very dependent on numerous factors, such as issues of model representation and measurement of association. Still, used with these problems in mind, it can provide highly improved performance, as compared a system without these techniques.

Hierarchical clustering:

One class of clustering algorithms generates hierarchical output. The hierarchy of clusters usually reflects more abstract concepts in the higher levels, and more detailed specific items in the lower levels.

Thesaurus generation:

Automatically generated thesauri contain classes that reflect the use of words in the corpus that is being examined. These classes do not naturally have a name, but are just a group of statistically similar terms. The optimum technique for generating the classes requires intensive computation. Other techniques starting with existing clusters, such as a basic manually created thesaurus, can reduce the computation required but may not produce optimum classes for that specific corpus.

Homonym problem:

When using automatic clustering approaches, it is very hard to eliminate homonyms that produce false hits when modeling the document relationships. With current techniques, humans have to perform this task. A homonym is when a term has multiple, different meanings (e.g., the term filed meaning an area of grass or an electromagnetic field). The longer (more terms in) the search statement, the less important is the human intervention to eliminate homonyms (Kowalski 1997). This is because items identified by the wrong interpretation of the homonym should have a low weight because the other search terms are not likely to be found in the item. When queries are short, significant decreases in precision will occur of homonym pruning is not applied. This indicates that “long” queries ought to be used whenever possible while searching, e.g. queries based on one or several texts describing a subject, as opposed to a few keywords.

Document clustering:

Clustering of items or documents is very similar to term clustering for the generation of thesauri. Manual item clustering is inherent in any library or filing system. In this case, a person reads the document (book, article, etc) and determines the category or categories to which it should belong. With physical clustering, such as with physical books in a library, each item is usually assigned to one category. This is also known as “hard” or Boolean clustering, or partitioning, as opposed to “soft” or fuzzy clustering, where each item can belong to several categories with some probability or weight factor. This idea is also used with many indexing systems. An item is physically stored in a primary category, but can be found in other categories as defined by the index terms assigned to the item.

Divisive algorithms start with one universal class to which all items belong, and each iteration in the clustering process involves choosing one of the current set of classes to split into two new classes. Agglomerative algorithms, in contrast, begin with each item belonging to its own class, then, in each iteration step, some pair of current classes is merged to form a new, larger class.

But there are also disadvantages. Hierarchical clustering suffers from the defect that it can never repair what was done in previous steps. Once an agglomerative algorithm has joined two objects, they can’t be separated. Also, whatever a divisive algorithm has split up cannot be reunited. The rigidity of hierarchical methods is both the key to their success (because it leads to small computation times) and their main disadvantage (the inability to correct erroneous decisions) (Lee 1997).

6 Results

In this chapter we will present our system design, and describe how we have implemented it. We start out with a brief idea of what we want to accomplish, followed by a step-by-step presentation of our implementation, with a short discussion about each step. In addition to this, an informal evaluation is presented.

6.1 Conceptual design implications

We have previously discussed different approaches to aid a user in web navigation whereof we have focused on automatic search engines (VISIT) and manually created link collection (IntraPages).

Building on these ideas, we propose a third way of supporting intranet navigation, which actually is trying to combine the most valuable features of the former two. We would like to combine the automatic approach taken by search engines, with the browsing capabilities and the organization of the hierarchical link collection.

To accomplish this, we mainly build on the same ideas as the search engines, i.e. web robots for data gathering, and automatic indexing based on research in Information Retrieval. This includes building document representations for the purpose of comparison. In a search engine, the query is transformed into a representation that can be compared with the document representations, and possibly ranked in some way. In our approach, formalized queries do not exist, so there is no need for representing them. Instead, we want to automatically generate a hierarchical index, resembling the manually created index. To accomplish this we use clustering techniques applied to the document representations.

6.2 Technical description of our system

We have chosen to implement an unsupervised method for indexing the documents of the intranet. We have chosen not to explicitly describe a given number of categories, instead we are building a symbolic tree, which describes the document inter-relationships. This tree can be viewed at different levels of granularity, and with different starting points, i.e. root nodes. This tree is way too big to be viewed at one time, it is just too much too look at. Our solution to this problem is to start at a given (root) node, and describe the closest branches as clusters. When an interesting cluster, or branch, is found, the user is able to zoom in at this branch and see more details about it, in the same way as above.

This process is iterated until the document, or leaf, level is reached and hopefully some valuable information is discovered. Each cluster is represented with a set of typical terms and pointers (URL's) to typical documents on the intranet, as well as numerical attributes. The number of documents in each cluster, and the tree depth is presented for each cluster. The "typical" terms and documents are the ones that are closest in angle in the multi-dimensional vector space, as described in the Latent Semantic Indexing section.

6.2.1 The Harvest system

To start with the intranet indexing tool builds an index, using the Harvest system, as we mentioned earlier, particularly the component called the Gatherer. This program is configured as a standard web robot, as described in section 4. Given a few webserver addresses to start with, it recursively follows all links it can find within the *volvo.se* domain, and stores information about all documents found. For this purpose, a format called SOIF (Summary Object Interchange Format) is utilized, which contains attributes as the URL (Universal Resource Locator), document type, document length, extracted ASCII text, etc. The Gatherer has functions to parse many different document types and convert these to plain (ASCII) text. Among these are well-known document formats such as HTML, PDF, PostScript, RTF, etc. There also exist summarizers for Microsoft Word as well as other common word processors. The Gatherer creates one SOIF file for each document found during the intranet search. All these files are stored in a directory mainly intended to be used the other main part of the Harvest system, namely the Broker, a search engine with a web-based front-end. At the Volvo intranet this component is known as VISIT.

6.2.2 Parsing the Harvest index structure

Our system begins its work after the Gatherer has run through the intranet and created its indexing structure. It starts with parsing the full SOIF directory, extracting all information we regard as interesting and useful, while disregarding some attributes, such as time-to-live and keywords extracted by the Gatherer system. These keywords, used by the Broker search engine, are not interesting because the Gatherer uses a much less sophisticated technique than we do, based on simple statistics. We prefer to base our analysis on the full text of each document. All the extracted information is saved in a couple of files for further processing in the next step. For future compatibility with other intranet index tools than the Harvest system, our parser is easy to modify to fit with another system. All that is needed is some knowledge of how to get the wanted information out of the specific index tool, for example where a certain directory with data files is located, and how these files are constructed.

6.2.3 Stripping non-alphabetical characters

Now we have direct access to the full (8-bit ASCII) text of all accessible intranet documents. The next step is to refine the text, by removing non-alphabetical characters. At this point, we have chosen to omit all numbers from the text. This may be changed in another run and the results compared to each other. All these non-alphabetical characters are substituted with space characters. In addition, all remaining characters are converted to upper case. A known problem at this point is how to handle non-English characters, such as the Swedish å, ä and ö. Many different systems are connected to Volvo’s intranet, and many of these have different ways of representing these characters. One widely used standard is ISO 8859-1, which is supported by many operating systems, and we use this one as well. But when it comes to texts written using for example MS-DOS standards, these have a different way of representing the å,ä,ö characters, and there is no sure way to handle this problem that we know of, from an intranet point of view. This problem originates in the old 7-bit ASCII standard, which only specifies the English character set a-z and A-Z.

6.2.4 Creating the word list

When dealing with document files, we have to decide how to treat and define single words. In our analysis, words are the least units (atoms) that make up a continuous text. We define a word as a consecutive string of alphabetical characters, including all national variants, as defined in the ISO-8859-1 (extended ASCII) standard, but not any other characters, such as space, tab, punctuation and numbers. To find each word, we scan the text stream until an alphabetical character is found. At this point a word is considered to begin. Now we continue the scanning until a non-alphabetical character is found, which marks the end of the word. This process is repeated throughout the entire text of all documents in the intranet corpus to generate a global frequency list. Other approaches to "word" definition exist, for example where words or sentences are broken down into consecutive strings of fixed length, often called n-grams, which may include parts of one or several "normal" words, as well as the spaces in between.

At this point the processed text is passed to a program which creates a global frequency list, i.e. a list of all instances of words in the entire corpus, together with the corresponding number of occurrences (word frequencies). All uppercase letters are converted into lowercase before this process begins. This list is sorted in frequency order, with the highest frequency first. In our test runs, this step yielded a list of some 130,000 unique words.

Figure 1. Frequency list with the 36 most common words on the Volvo intranet

49994	volvo	12371	nr	10343	key
31256	data	12325	message	10191	application
30706	file	12255	id	10123	set
28565	server	11615	files	10104	statement
19979	information	11409	vd	10076	database
17081	access	11363	network	9840	description
16855	internet	11126	vtc	9556	group
15355	program	11092	type	9554	class
15309	user	11054	memo	9477	control
15132	rfc	11027	end	9313	connected
14606	page	10767	number	9239	table
13004	command	10746	web	9121	directory

As you can see in the table above, not very surprisingly, the most common word in the document corpus is *Volvo*, with nearly 50,000 hits. By looking at the other top-36 words, we can conclude that much of the material available at Volvo intranet is orientated towards computer technology, in particular web technology. This is not a very surprising conclusion, though.

6.2.5 Stop-word filtering

Now we are faced with another problem: syncategorematic words, i.e. words that have no contextual meaning, such as conjunctions, prepositions, pronouns, etc, in both English and

Swedish (van Rijsbergen 1979; Salton 1989). These word are just in the way when we try to model the contextual or semantic relationships between the documents, since they do not add any meaning whatsoever to a piece of text in our model. All they do, if they are left in the text, is to slow down the rest of the process.

The actual stop list we use has been merged from different sources. The Swedish part comes from a research project at the Department of Linguistics at Gothenburg University. As you can see in table 3.2 that there are some misspelled words included in the list, e.g. the correct spelled AMONGST and the misspelled AMOUNGST. Through statistically done surveys it is proven what words people have problem to spell correctly. The English stop words came from the book "Information Retrieval" by van Rijsbergen.

Table 1. An abstraction of the stopword-list

A	AKTUELLA	ALLOWS	ALREADY
AA	AKTUELLT	ALLRA	ALRIG
AAH	ALDRI	ALLSÅ	ALLRI
AB	ALDRIG	ALLT	ALLS
ABOUT	ALL	ALLTFÖR	ALSO
ABOVE	ALLA	ALLTI	ALTHOUGH
ABSOLUT	ALLAS	ALLTID	ALWAYS
ACROSS	ALLDELES	ALLTIHOP	AM
AE	ALLDES	ALLTIHOPA	AMONG
AFTER	ALLDRIG	ALLTING	AMONGST
AFTERWARDS	ALLE	ALLTMER	AMOUNGST
AGAIN	ALLIHOP	ALLTSÅ	AMOUNT
AGAINST	ALLIHOPA	ALMOST	AN
AH	ALLMÄNNA	ALONE	AND
AHA	ALLMÄNT	ALONG	ANDRA

The global frequency list is simply matched with a list of stop-words, and all co-occurrences result in the removal of that word from the frequency list. This process usually removes some 3000 words or so, but since most of these words are relatively common (high frequencies), the total number of removed words from the corpus is much greater.

We made some minor additions with words that we found specific for Volvo that did not add any value. For example in our case we added "Volvo" in the list we used, because it is the organization's name and therefore is represented in almost every single document. Also we added the html-tag NBSP, that is included on almost every row in every document but that is not stripped away with all the other html-tags, because it is not omitted with < and >.

6.2.6 Stemming

The next refinement is performed with a technique known as stemming, i.e. trying to group words by their common stem. Looking for common suffixes, as well as rules for matching similar pronunciations, and other rules, achieve this. The output of this step is a file describing found relationships between the words from the frequency list, according to a set of rules for the stemming program. A new frequency list is constructed using this information, where all words considered to be closely related to each other are grouped as one single entry in the new frequency list, with all their individual frequencies summed. During this process, about 25% of all the original words are eliminated, shortening the frequency list to three quarters.

Table 2. An example of word groups created by the stemming program

ABSORB	ABSORBER	ABSORBERS	ABSORBS	ABSORBERAR	ABSORBERAS
ABSTRACT	ABSTRACTS	ABSTRAKT	ABSTRAKTA		
ACCELERATED	ACCELERATE	ACCELERATES			
ACCELERATOR	ACCELERATORS	ACCELLERATOR			
ACCELERERAD	ACCELERERA	ACCELERERAT			
ACCENTUERATS	ACCENTUERAR	ACCENTUERADES			
ACCEPT	ACCEPTEN	ACCEPTS			
ACCEPTANCE	ACCEPTANCES	ACCEPTENCE			
ACCEPTERA	ACCEPTERAD	ACCEPTERADE	ACCEPTERADES	ACCEPTERAR	ACCEPTERAS
ACCOMMODATE	ACCOMMODATED	ACCOMMODATES	ACCOMODATE	ACCOMODATES	
ACCOMMODATION	ACCOMMODATIONS	ACCOMODATION	ACCOMODATIONS		
ACKNOWLEDGED	ACKNOWLEDGE	ACKNOWLEDG	ACKNOWLEDGER	ACKNOWLEDGES	
ACKNOWLEDGMENT	ACKNOWLEDGEMENTS	ACKNOWLEDGEMENT	ACKNOWLEDGMENTS		
ADMINISTRATE	ADMINISTRATED	ADMINISTRATES			
ADMINISTRATION	ADMINISTRATIONEN	ADMINISTRATIONS			
ADMINISTRATIVE	ADMINISTRATIV	ADMINISTRATIVA	ADMINISTRATIVT		
ADMINISTRATÖR	ADMINISTRATÖREN	ADMINISTRATÖRER	ADMINISTRATÖRERNA		
ADMINISTRERAS	ADMINISTRERA	ADMINISTRERAD	ADMINISTRERADE	ADMINISTRERAR	
ADRESSEN	ADRESSER	ADRESSERNA	ADRESSED	ADDRESSED	ADRESSERA
ADRESSERAD	ADRESSERAS	ADRESSERAT			

We are using a stemming program that has been developed for the purpose of research in linguistics. It has previously been used in a study at Apoteksbolaget, a Swedish government agency in the pharmaceutical domain. This study concluded that about 90% of the suggestions for word grouping according to stem were accurate, which is a very good result (Grönqvist 1997). The stemming algorithm is optimized for the Swedish language, which means it does not perform perfectly when applied to English, but still it gives quite a good result. There is one remaining problem that can not be solved by automatic routines. Names can be spelled in similar ways or even the very same name can refer to different persons. This name-relationship does not deduce that these persons have something else in common and is thus inappropriate for our document relationship model. In other terms, syntactic relationships does not always deduce semantic relationships, but in the case of common stems, it does.

6.2.7 Further refinement

After the stemming algorithm has been applied, words that appear only once in the remaining frequency list are now disregarded, because they only appear in a single document, without even a closely related word somewhere else. This is yet another step towards a small and compact model of semantic relationships between documents. A great number of words are removed at this point. During our test runs, another 35% of the words are eliminated from the frequency list. In addition to this, words with a length less than three characters, i.e. words with only one or two characters are also removed. These words are too short to carry any reliable information about syntactic or semantic relationships, see the discussion about personal names above. Not too many words are removed in this process though, only about 0.7%.

At this point, we have created a set of words, contained in the remaining frequency list, describing the main contextual or semantic relationships between the documents in our collection, with related frequency for each word. If these words are regarded as a set of words rather than an ordered list, this set constitutes the terms we use for creating our mathematical model.

6.2.8 Onto the documents themselves

Now, each single document is processed once again, compared with the set of terms, local (within the document itself) frequency of all relevant words is calculated. Each term is associated with a specific term weight, according to scheme known as log-entropy weighting. This scheme takes into account both the global and local frequency of the terms, as well as how many documents they appear in. A term database is created where each term and its associated term weight, a real number in the interval [0,1] is stored. The higher the weight, the more “important” is that single term, in some sense (hard to describe explicitly!).

6.2.9 Building the matrix

Now we are ready to create the actual mathematical model, applying the vector space metaphor. We construct a matrix where the rows consist of term vectors, and columns of document vectors. The local frequency of each term in each document is entered into the document vector, and all non-zero term elements are multiplied with their associated weight. As we are dealing with quite a large set of terms, and most of the documents are quite short as well, resulting in a very sparse matrix. In our test runs, about 0.1 % of the elements in the matrix were occupied by non-zero elements.

6.2.10 Singular Value Decomposition and LSI

In theory, the document vectors in our matrix could be clustered as is. There is one overwhelming problem though. The sparseness of the matrix makes it very difficult to distinguish between subtle differences between documents. Another problem is calculation time. In order to construct a usable system, it would be unrealistic to perform clustering (using any clustering technique) with 50,000-dimensional vectors. This would simply take too long time! One approach in trying to overcome these problems is to drastically reduce the dimensionality, using a linear algebra method known as Singular Value Decomposition (SVD). This approach, called Latent Semantic Indexing (LSI) was first developed by (Deerwester, Dumais et al. 1990), and has since then been further refined and developed. LSI is described in more detail in chapter 5. Applying SVD to the large, sparse matrix, produces three new matrices, which when multiplied reconstructs the original matrix. However, the middle matrix of these three matrices is a diagonal matrix with singular values, the higher the singular value, the more “important” is that single relationship associated with the singular value and its accompanying document and term vectors. The 300 or so most important relations are selected, and the rest of the matrices are disregarded. Through this truncation, we get new document vectors, with much more compact information about the inter-document relationships. Another benefit is that even more “noise” seems to be reduced, according to empirical evaluations (Dumais 1995). Also, all the new vectors are orthogonal, which improves the clustering results even more, since the vectors are linear independent of each other.

6.2.11 Clustering

Finally, we are ready to perform the actual clustering. In our implementation, we have chosen to use a technique known as Hierarchical Agglomerative Clustering. We have implemented a program in C that takes the document vectors as input and produces a hierarchical tree structure of the inter-relationships between the document vectors. This structure is saved in a file that is used by the browsing program. There is however a major drawback with this program, since it takes rather long time to execute on large datasets. The program itself could possibly be optimized in some way to accomplish faster clustering, but the real limitation is the algorithm.

6.2.12 Browsing the clustered index

Now that the final part of the automatic indexing is done, the user can take over and view the results. We have implemented a program that gives the possibility to browse the tree structure that was created in the clustering phase. This program starts at a specified node, initially the main root node, and displays parts of the tree in the form of clusters. The current viewed tree is searched through and divided into about 8 subclusters, depending on the actual tree structure. These clusters consist of the most significant branches from that node according to some criteria (relative number of leafs on the tree branch, branch depth, etc). For each subcluster (branch), a number of characteristics are displayed: typical words, typical documents, the relative size, and the number of included documents. Typical words and documents are computed in the following way: For each subcluster, its so-called centroid vector is calculated as the arithmetical average vector of all document vectors within that subcluster. This centroid vector is considered to express the ‘essence’ of the subcluster in some way, and is represented in the same vector space as all other vectors. The 10 term and document vectors that have the smallest angle to the centroid vector are taken as being representative of that subcluster, and are being displayed on the screen. We have though about other means of representation a subcluster, such as the most highly frequented words within each subcluster. However, this would require individual frequency lists of all documents to be available at this time, which would take up a huge amount of disk space, not to mention the time it would take to compute these merged frequency lists for each iteration during the browsing.

We tried to implement a web interface, but we did not get it to work the way we wanted, because of long response time, and problems with getting the programs to work within the Apache web server and its CGI-interface. For instance, the CGI programs would run as processes owned by the user ‘nobody’, and that user has severe limitations to its available actions, how much memory it may allocate for a process, etc. We would have had to spend too much time with configuring Unix systems and web servers. It just did not seam reasonable to put this much effort into doing this.

6.3 Implementation details

All parts of the system were developed on a Unix platform running Sun Solaris. We have used Perl, C and Shell scripts when programming the actual implementation. In the beginning of the project we spent quite some time at Volvo, but then we were offered a room at the department with some suitable development facilities. As time went by, we spent the major time at the Department of Informatics, and the programs were then transferred to Volvo for testing and evaluation.

6.4 Evaluation

An informal evaluation has been conducted with Dick Stenmark and Martin Börjesson from Volvo. They have both been working with the intranet in many ways for years, and are among the most experienced experts at Volvo in this matter. Therefore it seemed natural to ask them for their opinions of our system, and how well the results thereof corresponds to their conceptual view of the Volvo intranet.

We conducted the evaluation with them, one at a time, when they were presented to the browsing interface of the document tree, as well as other parts of the system, e.g. frequency list output and stemming tables. This was mainly done in a unix environment, and sometimes through a web interface. We provided the experts with brief instructions of how to use the interface, and how the output should be interpreted. Then we sat in a group in front of a computer and went through the utilities, while commenting and discussing the results. This type of evaluation was done at several stages throughout the development cycle.

The overall opinion was that the system could find many interesting connections and indeed provided some insights in the structure of the intranet. However, some of the arrangements that were displayed on the screen were hard to interpret, since there seemed to be little or no logical connection between the documents that the system had chosen to put in the same cluster in some cases. This is of course a serious problem, which corresponds to what other critics of LSI have said. Still, LSI is considered as one of the most prominent ways of modeling document relationships (Dumais 1995), when compared to other available approaches. No automatic indexing system has been perfect so far.

7 Discussion

An additional viewpoint to the performance of our system is given by a related project conducted by our fellow students Henrik Högberg and Anders Gustafson. They have partially based their master thesis project on our work, and have applied Genetic Algorithms trying to model relevance feedback for a user's search profile. In their work, they use our automatic text indexing system to create vectors that represent available documents. A user's interest profile is modeled in the same representation space, and they apply the same metrics for comparison between documents as we do. Their studies indicate that Genetic Algorithms are applicable in this context, and they have some promising results.

However, all of their work is based on the assumption that the vectors from our indexing system really provide a valid representation of the document inter-relationships. And since the rest of their system seems to work quite well, implying that our modeling scheme turns out to be decent. Maybe we just have to reconsider the clustering algorithms, since it seems that our document representations are accurate.

7.1 Future work

Utilize metadata available from HTML, PDF, MS Word, and other information sources that are semi-structured.

Combine the ideas presented in this thesis with autonomous agent technology, to form awareness and CSCW applications (Ljungstrand and Fagrell 1998).

We should take advantage of heuristics in the context of how web documents are organized. What should be considered as one item or document? A single file? How can the inherent hypertext structure be exploited? What about local directory structures on the web server and other domain specific knowledge?

7.2 Conclusions

We believe that:

Linguistic methods can be combined with the statistical ones (such as the LSI-model) for synergy effects.

Integrating the best parts and pieces from diverse disciplines and theories, for a specific situation, is a good idea, but difficult to master.

There is a great potential benefit in constructing and using systems for automatic analysis and "knowledge" extraction of corporate intranets.

This thesis is pointing in the right direction, but there is still a lot of work and research to be done.

Text-based analysis such as ours, can lead to many other interesting applications, where the information gained by our system can be used as an input in for example recommender systems, etc higher level organizational descriptive information or model.

Clustering is a powerful technique that we have found useful when dealing with large quantities of data, represented by a formal model, but it is hard to master.

8 Appendices

8.1 Appendix A – Mathematical overview of LSI

The reduced document representation in LSI has the following advantages to the original vector-space representation (Deerwester, Dumais et al. 1990):

- the dimensions in the space are uncorrelated (i.e. they are orthogonal)
- the representations are less noisy
- the representations incorporate higher-order (latent) association structure among terms and documents
- the information is stored a much more compact and efficient way, which means less requirements for memory and processing speed after the SVD has been performed

These properties are a result of Singular Value Decomposition (SVD), a technique from the Linear Algebra field of mathematics. A more mathematical presentation of the method is presented below. This part may be skipped without loss of understanding of the effects of SVD and LSI. Refer to any book on Linear Algebra and Numeric Analysis, such as (Petersson 1993) or (Starius 1996), for a definition of the mathematical terms discussed below.

Suppose **X** is an arbitrarily matrix with t rows and d columns, denoted $t \times d$, and rank r. SVD is a technique for uniquely decomposing the matrix **X** as the product of three matrices:

$$\mathbf{X}=\mathbf{ULA}^T \qquad (1)$$

where $\mathbf{A}^T$denotes the transpose of **A**. Special restrictions apply to these matrices: **U** and **A** are both column orthogonal and have unity length, i.e. they are orthonormal. Thus, $\mathbf{U}^T\mathbf{U}=\mathbf{I}$, where **I** is the identity matrix, and $\mathbf{UU}^T=\mathbf{P}$, where **P** is a projection matrix on to d-dimensional space. **L** is a diagonal matrix of *singular values:* all non-diagonal elements are zero; all diagonal elements (the singular values) are non-negative real values, typically ordered by monotonically decreasing value. **U** is allowed to be $t \times r$, **A** to be $d \times r$, and **L** to be $r \times r$, with no zero singular values.

SVD provides the best lower rank approximation of a matrix **X** in terms of the Euclidean matrix norm (2-norm), i.e. in the least squares sense, due to a theorem by Eckart and Young (Letsche and Berry 1997). More formally, let $\mathbf{U}_k$ be the $t \times k$ ($k \le r$) matrix found by removing $r - k$ columns from **U**. The k columns remaining in $\mathbf{U}_k$ correspond to the largest singular values in **L** (similar versions of $\mathbf{L}_k$ and $\mathbf{A}_k$ can be defined). Then $\mathbf{X}_k = \mathbf{U}_k\ \mathbf{L}_k\ \mathbf{A}_k{}^T$ minimizes $\| \mathbf{X}_k - \mathbf{X} \|_2$ over all rank-k $\mathbf{X}_k$, where $\| \cdot \|_2$ denotes the Euclidean matrix norm (calculated by taking the square root of the sum of all squared entries of a matrix).

Now, what can we use this for? In LSI, we use SVD to derive the uncorrelated and reduced dimension representation of the documents. Suppose **X** ($t \times d$) is a matrix of d documents represented using t terms, and $\mathbf{X}=\mathbf{ULA}^T$ is the singular value decomposition of **X**, then row i of $\mathbf{A}_k\mathbf{L}_k$ gives the representation of document i in k-space. These re-representations of the documents are used in place of the original t-space representations when measuring similarity between documents. Reduced term vectors can be defined in a similar manner. If queries used for information retrieval are represented as vectors in the original term space, then these queries can also be mapped on to the reduced *k*-dimensional document space, and used for similarity judgments.

Another important issue is how many dimensions should be removed, and how many should be kept? Experiments have shown that although a high-dimensional representation appears to be required for good retrieval performance. Care must be taken not to reconstruct the original matrix **X**. If **X** is nearly reconstructed, the noise caused by variability of word choice and terms that span or nearly span the document collection won't be eliminated, resulting in poor performance. Empirical studies suggest keeping about 50 to 300 dimensions, out of, in most cases, several thousands or tens of thousands (Berry, Dumais et al. 1995).

Two vectors *A* and *B*, in three dimensions:

$$A = [a_1, a_2, a_3] \qquad B = [b_1, b_2, b_3]$$

The dot or inner product of *A* and *B*:

$$A \cdot B = [a_1, a_2, a_3] \cdot [b_1, b_2, b_3] = a_1 b_1 + a_2 b_2 + a_3 b_3$$

The length of a vector *A*:

$$|A| = |[a_1, a_2, a_3]| = \sqrt{{a_1}^2 + {a_2}^2 + {a_3}^2}$$

The angle between A and B can be calculated:

$$\cos\alpha = \frac{A \cdot B}{|A||B|}$$

8.2 Appendix B – Common similarity measures

Table 1.2: Commonly used measures of association.

Inner product (dot) measure:	$M(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} x_i y_i$
Cosine measure:	$M(\mathbf{X}, \mathbf{Y}) = \dfrac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}}$
Manhattan distance measure:	$M(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} \lvert x_i - y_i \rvert$
Euclidean distance measure (2-norm):	$M(\mathbf{X}, \mathbf{Y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$
m-norm measure:	$M(\mathbf{X}, \mathbf{Y}) = \left(\sum_{i=1}^{n} (x_i - y_i)^m \right)^{\frac{1}{m}}, \; m \in \boldsymbol{N}$

Where $\mathbf{X} = (x_1,x_2,...,x_n)$ and $\mathbf{Y} = (y_1,y_2,...,y_n)$ are two *n*-dimensional vectors.

8.3 Appendix C – Common weighting functions

Table 1.3: Common local term weighting functions.

Local weighting functions $L(i, j)$:

Binary: $L(i, j) = 0$ if $tf_{ij} = 0$

$L(i, j) = 1$ if $tf_{ij} > 0$

Term-Frequency: $L(i, j) = tf_{ij}$

Log: $L(i, j) = \log_2(tf_{ij} + 1)$

Where:

tf_{ij} = the frequency of term i in document j

Table 1.4: Common global term weighting functions.

Global weighting functions $G(i)$:

Normal: $G(i) = \sqrt{\dfrac{1}{\sum_j (tf_{ij})^2}}$

Gf-Idf: $G(i) = \dfrac{gf_i}{df_i}$

Idf: $G(i) = \log_2\left(\dfrac{ndocs}{df_i}\right) + 1$

Entropy: $G(i) = 1 - \sum_j \dfrac{p_{ij}\log_2(p_{ij})}{\log_2(ndocs)}$, $p_{ij} = \dfrac{tf_{ij}}{gf_i}$

Where:

tf_{ij} = the frequency of term i in document j

gf_i = the global frequency of term i

df_i = the number of documents in which term i appears

$ndocs$ = the number of documents in the collection

8.4 Appendix D – Harvest overview

Harvest is an integrated set of tools to gather, extract, organize, search, cache and replicate relevant information across the Internet (Hardy et. al. 1996). It is developed at the Department of Computer Science at the University of Colorado, at Boulder. With no big means, a user can tailor Harvest to collect information in many different formats, and also offer custom search service on the Internet. One primary objective is to provide a system that can be configured different ways to be able to create indexes. Harvest can also make very efficient use of Internet servers, network links, and index space on disk.

Harvest also makes it possible for users to extract structured information from many different formats and build indexes that allow these attributes to be referenced during queries e.g., searching for all documents with a certain regular expression in the title field. An important advantage of Harvest (no, we are not getting paid for saying this! ;) is that it provides a data gathering architecture for constructing indexes. It allows users to build indexes using either manually constructed templates for controlling the index content that gives maximum control over the collected data, or automatically extracted data constructed templates, which would make it easy to cover large data collections. A third way is to use a combination of the two methods.

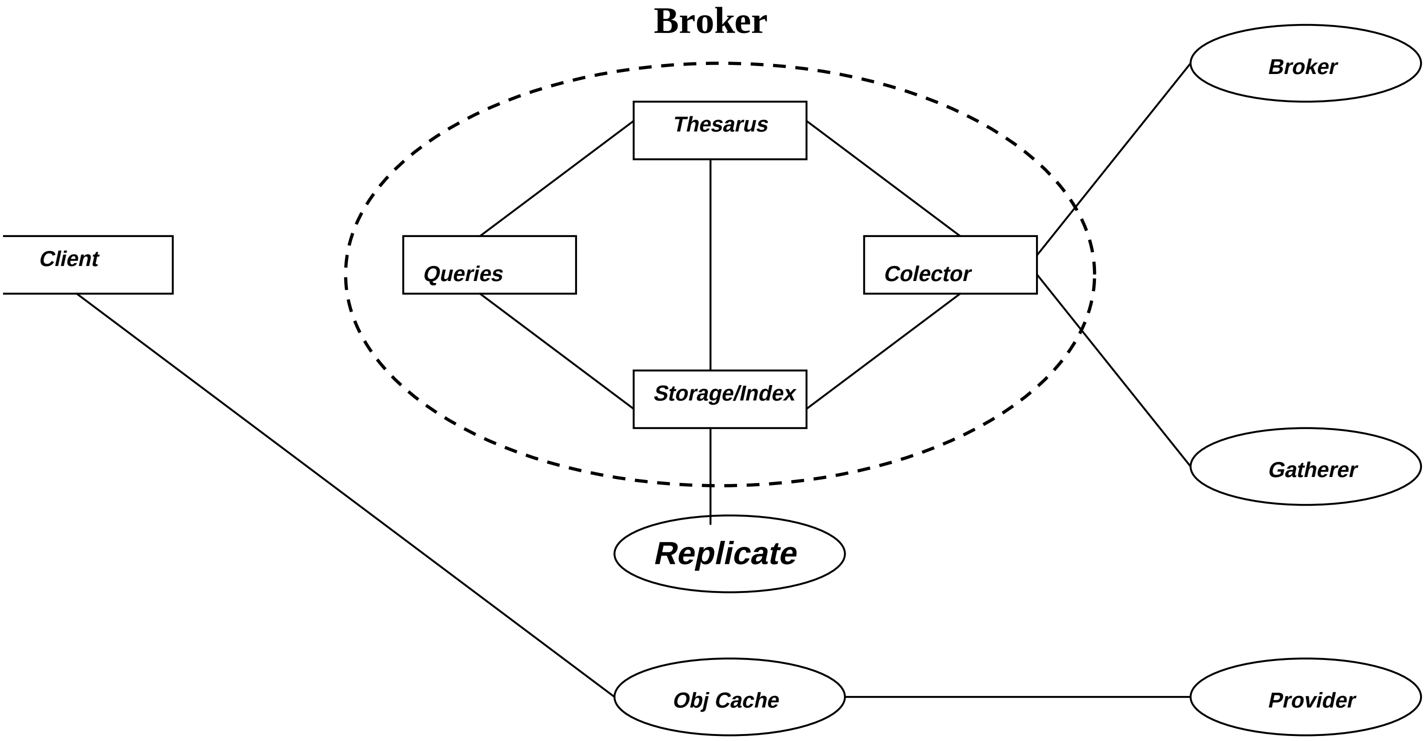


Figure 1: Overview of Harvest Software Components.

Harvest consists of several subsystems. First, the Gatherer subsystem collects indexing information, such as keywords, author names, and titles, from the resources available at Provider sites like http severs. The Broker subsystem retrieves indexing information from one or more Gatherers eliminating duplicate information. The Broker also incrementally indexes the collected information and provides a WWW query interface to it. A user can efficiently retrieve located information through the Cache subsystem.

Appendix E – Related applications and systems

Here we will briefly look at some other systems that have tried to address problems similar to ours.

8.4.1 The SMART Information Retrieval System

The SMART information retrieval system, initiated by Salton *et. al.* at Harvard University in 1964 and continued at Cornell University in 1968 (Frakes and Baeza-Yates 1992), embodied many of the information retrieval techniques found in modern vector-space systems. By using stemming to remove suffices from terms, and a variety of dictionaries and thesauri to increase the effectiveness of the many different information analysis techniques incorporated in SMART, SMART became an important research vehicle as well as an effective information retrieval system (Salton and Lesk 1965).

Salton and other researchers used SMART to study term-weighting, relevance feedback, clustering, stemming, synonyms, and the use of phrases to improve the performance of the retrieval system (Frakes and Baeza-Yates 1992). In particular, SMART used syntactic phrase matching to recognize similar concepts in the documents and to allow documents with similar concepts to be retrieved. It also used statistical phrase matching to detect the co-occurrence of concepts in sentences. SMART attempted to adapt to the query being performed by allowing the user to choose from a variety of automatic analysis procedures to process the query. In addition, the different analyses could be automatically compared to determine their relative performance and to ensure most of the documents matching the query were found (Salton and Lesk 1965).

8.4.2 Self-Organizing Maps

Kohonen *et. al.* in Finland. Websom (Honkela, Kaski et al. 1996).

See http://websom.hut.fi/ for more details, there is even an interactive demo of Websom applied to a corpus of Usenet articles.

If no keywords are available and the texts are very colloquial such as the free-form discussions in the Internet newsgroups are, new full-text searching methods have to be developed. Let us tentatively imagine that we first form word histograms from the different documents. Although the number of different words or other expressions used in an Internet newsgroup may be on the order of one hundred thousand, we could restrict to, say, 6000 words that are most common and most descriptive of the contents. Rare words can be discarded automatically. From the remaining vocabulary one can easily cancel non-descriptive words manually.

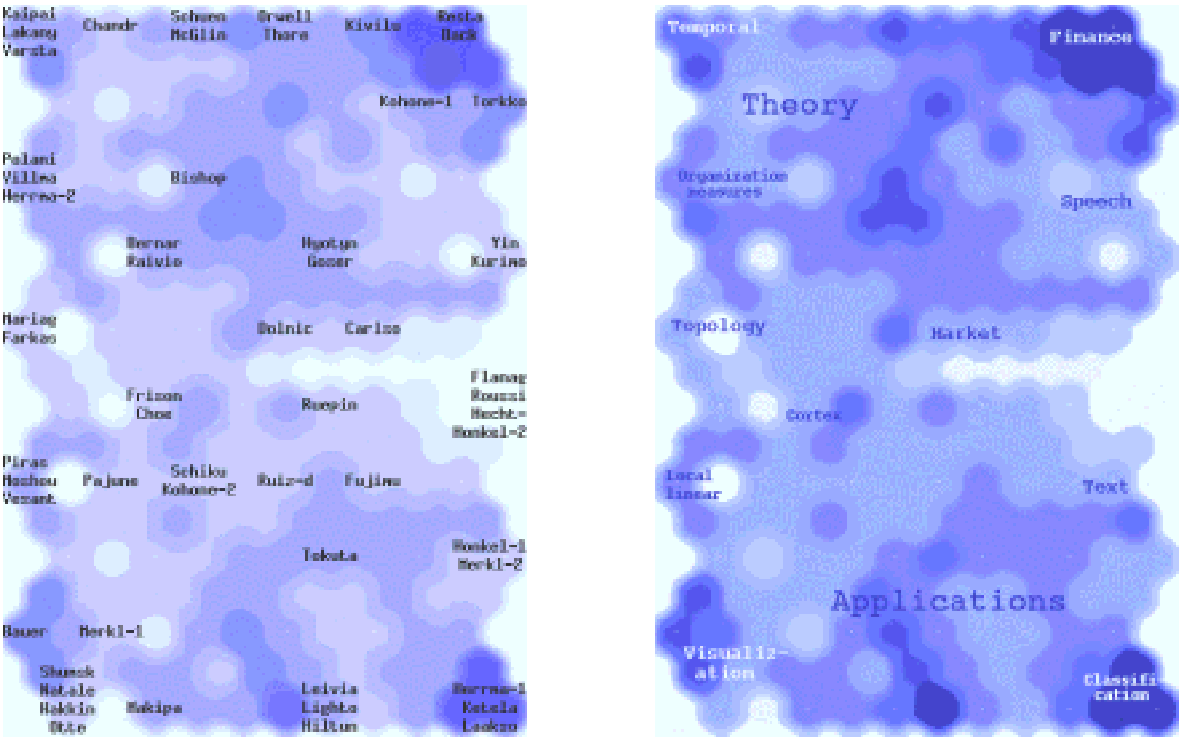


Figure 1. Pictures of Self-organizing maps, applied to the articles from WSOM'97 - Workshop on Self-Organizing Maps, Helsinki, Finland.

After that we aim at the representation of each document as a point on a two-dimensional display (the "map") in such a way that the mutual distance between any two representation points is roughly inversely proportional to the similarity of the corresponding two histograms. Therefore similar documents would become mapped close to each other on the map, like the books on the shelves of a well-organized library. In the SOM method we are actually not comparing the similarities between all pairs of histograms but the similarities of histograms with certain reference vectors (model vectors). The latter are then adaptively changed during the computing process to minimize certain estimation errors. The 6000-element histograms are still inconveniently large to deal with. Therefore it is desirable to be able to group the words into much fewer meaningful categories, and represent the documents as category histograms. No manual analysis or preparation of the texts, of course, should thereby be necessary. We have earlier used the SOM algorithm to study short segments of texts, such as triplets of successive words, and to cluster the words automatically on the basis of this contextual information. With this method we have now been able to reduce the size of the histograms to 315 elements. Moreover, these histograms are to a great extent invariant with respect to the choice of particular words, mainly characterizing what categories of terms and in what context are being used in the documents. When these word category histograms are used to form a map of the document collection, we can thus differentiate the various documents in an orderly fashion in this "document map", as a meaningful organizational structure that can be explored easily.

The overall "architecture" of the WEBSOM method is presented in Fig. 1. We have demonstrated its potential in case studies where articles from selected Usenet newsgroups were organized. The articles were colloquial, mostly rather carelessly written short discussions, frequently containing spelling errors. In the studies, the resulting order on the document map was found to reflect topical relations and differences between the newsgroup articles; similar articles tended to occur near each other, often in the same location of the map.

The self-organizing map as presented in (Kohonen 1995) is an unsupervised artificial neural network model. The model consists of a layer of inputs each of which is fully connected to a grid of output units. These output units are arranged in some topological order where a two-dimensional grid represents the most common choice.

Input units take the input pattern and propagate them as they are onto the output units. Each of the output units is assigned a weight vector with the same dimension as the input data.

The learning process of self-organizing maps can be seen as a generalization of competitive learning. The key idea is to adapt the unit with the highest activity level with respect to a randomly selected input pattern in a way to exhibit an even higher activity level with this very input in future. Commonly, the activity level of an output is computed as the Euclidean distance between the unit's weight vector and the actual input pattern. Hence, the so-called winning unit, i.e. the winner in short, is the output unit with the smallest distance between the two vectors. Adaptation takes place at each learning step and is performed as a gradual reduction of the difference between the respective components of input and weight vector. The degree of adaptation is guided by a so-called learning-rate that is gradually decreasing in the course of time.

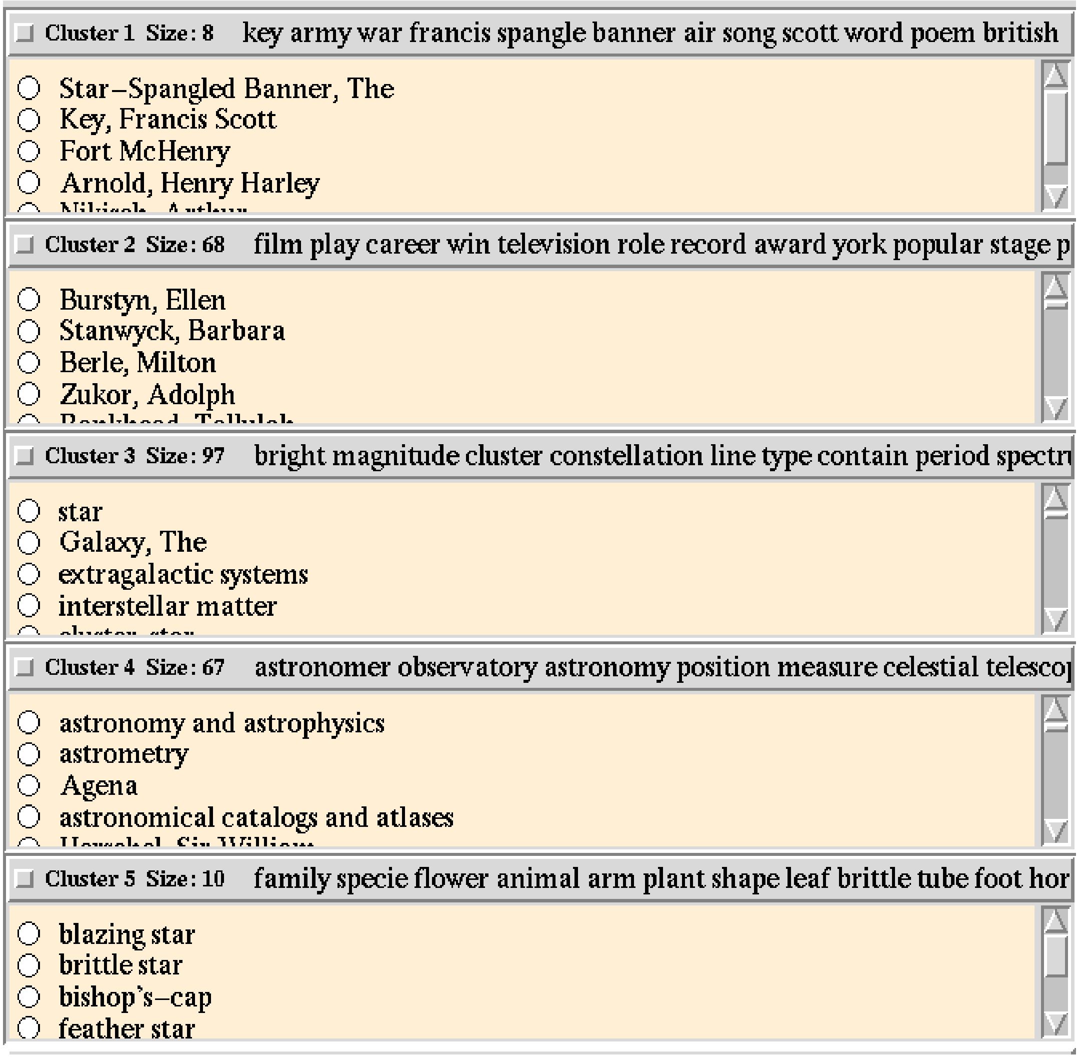
As an extension to competitive learning, units in a time-varying and gradually decreasing neighborhood around the winner are adapted, too. Pragmatically speaking, during the learning steps of self-organizing maps a set of units around the actual winner is tuned towards the currently presented input pattern. This learning rule leads to a clustering of highly similar input patterns in closely neighboring parts of the grid of output patterns. Thus, the learning process ends up with a topological ordering of the input patterns. One might say that self-organizing maps represents a spatially smooth neural variation of k-means clustering, where k is equal to the number of output units.

A variation the self-organizing maps used in Websom, trying to combine this model with a hierarchical approach, as been developed by Merkl *et. al.* (Merkl and Tjoa 1996). This model seems promising, since it is much more computationally effective, because the learning time is greatly reduced. It is based on small, hierarchically arranged unsupervised neural networks.

8.4.3 Scatter/Gather

Scatter/Gather is a cluster-based browsing technique for large text collections, developed at Xerox Palo Alto Research Center by Marti Hearst *et. al.* Here users are presented with automatically computed summaries of the contents of clusters of similar documents and provided with a method for navigating through these summaries at different levels of granularity. The aim of the technique is to communicate information about the topic structure of very large collections.

Scatter/Gather document browsing technique is aimed at supporting such exploratory learning. The emphasis in this browsing technique is to present users with an automatically computed overview of the contents of a document collection, and to provide a method for navigating through this summary at different levels of granularity. The central primitive operation in Scatter/Gather involves document clustering based on pairwise document similarity. The technique aims to place similar documents into the same cluster. Recursively clustering a collection produces a cluster hierarchy. For each cluster, at each level of this hierarchy, the user is presented with summary information about the cluster that presumably communicates something about the kinds of documents it contains. The user may then select (gather) those clusters that seem interesting or relevant. This subset of clusters can then be reclustered (scattered) to reveal more fine-grained clusters of documents. With each successive iteration of scattering and gathering clusters, the clusters become smaller and more detailed, eventually bottoming out at the level of individual documents.



□ Cluster 1 Size: 8 key army war francis spangle banner air song scott word poem british

○ Star–Spangled Banner, The
○ Key, Francis Scott
○ Fort McHenry
○ Arnold, Henry Harley

□ Cluster 2 Size: 68 film play career win television role record award york popular stage p

○ Burstyn, Ellen
○ Stanwyck, Barbara
○ Berle, Milton
○ Zukor, Adolph

□ Cluster 3 Size: 97 bright magnitude cluster constellation line type contain period spectr

○ star
○ Galaxy, The
○ extragalactic systems
○ interstellar matter

□ Cluster 4 Size: 67 astronomer observatory astronomy position measure celestial telesco

○ astronomy and astrophysics
○ astrometry
○ Agena
○ astronomical catalogs and atlases

□ Cluster 5 Size: 10 family specie flower animal arm plant shape leaf brittle tube foot hor

○ blazing star
○ brittle star
○ bishop's–cap
○ feather star

Figure 1. An example of a Scatter/Gather text clustering

The system scatters the collection into a small number of document groups, which is called clusters. It presents short summaries of the documents to the user and based on these summaries, the user selects one or more of the groups for further study. The selected groups are then gathered together to form a sub collection. Then Clustering once again is applied to scatter the new sub collection into a small number of document groups, which are again presented to the user. With each successive iteration the groups become smaller, and therefore more detailed. At the end of the iterations the groups will be small enough and the process have produced an enumerated list of individual documents.

Shown in figure 1 are the clusters' sizes, how many documents they contain, and a list of both topical terms and the document titles. One can see from the topical terms of Cluster 1 that this cluster contains documents that involves stars as symbols, as in military rank and patriotic songs. Cluster 2 has 68 documents that appear mainly to be about movie and TV stars. Cluster 3 contains 97 documents that having to do with aspects of astrophysics. Cluster 4 contains 67 documents also about astronomy and astrophysics. This cluster contains many articles about people who are astronomers (this is apparent when the list is scrolled down). Cluster 5 contains all the articles that discuss animals or plants, and that happen to contain the word star, for example, star fish.

When faced with ill-defined problems requiring information access, we often want to explore the resources available to us before exploiting them. This exploration may be partly aimed at refining our understanding of the potential space of content that is available, and partly aimed at formulating a concrete course of action for getting specific documents. Interfaces that support the browsing of a collection, as opposed to searching a collection, are aimed at satisfying this need to learn more about a collection before taking action.

8.4.4 Agent-based approaches

In the last few years, the rate of increase in information published on any media has been estimated to double every 20 months (Piatetsky-Shapiro & Frawley, 1991). The largest increase in information storage and communication, by far, has been on the Internet. For instance, the volume of Usenet News generated each day exceeded 100MBytes in 1994. The rate of Internet news traffic is doubling every year. The total Internet traffic has been increasing at an even faster pace of 12 percent per month, corresponding to a doubling of the traffic every six months (Witten, Moffat & Bell, 1994).

The introduction of the World Wide Web (WWW) has been primarily responsible for the explosive growth in Internet publishing and communication. During 1994 alone, WWW traffic has been estimated to increase more than 15 times (Quarterman, 1995). This phenomenal increase in the information published on the WWW, while providing for information dissemination, also makes finding relevant material in this sea of information a great challenge.

Today the typical user experience on the Web is through surfing - navigating through the Web space by following hyperlinks. While the dominant Web usage is the direct manipulation method (i.e. surfing), the following underlying characteristics of the Web environment dictate why we need Internet agents for information brokering.

The volume of information on the Internet is huge, currently served by approximately thirteen million hosts and is getting larger, the number of hosts doubling every year. The type of information on the Internet varies widely from newsgroups to corporate public relations, from personal position papers to academic journal articles. The quality of information has a large variance, information-rich documents in addition to poor quality material, as there is no control on what gets published on the Internet.

The depth-first surfing inherently encouraged by Web browsers causes most users to get lost in Web hyperspace. Given that the Web has grown immensely beyond its original homogeneous origin serving mainly high-energy physics researchers, it is now practically impossible for a Web user to find all of her/his information interests through surfing. Furthermore, the rich knowledge sources on the Web also make it extremely hard for mainstream users to know what, where, and how to find the right information. Intelligent agents promise to address these user needs on the Web.

9 Bibliography

Bartell, B., G. Cottrell, et al. (1992). Latent Semantic Indexing is an optimal Special Case of Multidimensional Scaling. 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.

Belkin, N. J. and W. B. Croft (1992). “Information Filtering and Information Retrieval: Two sides of the same coin?” Communications of the ACM **35**(12): 29-38.

Berry, M. W., S. T. Dumais, et al. (1995). “Using Linear Algebra for Intelligent Information Retrieval.” SIAM Review **37**(4): 573-595.

Brin, S. and L. Page (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. Seventh International World Wide Web Conference (WWW7), Brisbane, Australia.

Bush, V. (1945). “As We May Think.” The Atlantic Monthly **176**(July): 101-108.

Deerwester, S., S. Dumais, et al. (1990). “Indexing by latent semantic analysis.” Journal of the American Society for Information Science **41**(6): 391-407.

Dumais, S. T. (1995). Using LSI for information filtering: TREC-3 experiments. The Third Text REtrieval Conference (TREC3), National Institute of Standards and Technology Special Publication.

Eichmann, D. (1994). Ethical Web Agents. The 2nd International World Wide Web Conference, Chicago, USA.

Frakes, W. and R. Baeza-Yates, Eds. (1992). Informatin Retrieval: Data Structures & Algoritms. Englewood Cliffs, New Jersey, Prentice Hall.

Furnas, G. W., T. K. Landauer, et al. (1987). “The vocabulary problem in human-system communications.” Communications of the ACM **30**(11): 964-971.

Grönqvist, L. (1997). Personal communication, Department of Lingustics, Göteborg University.

Hardy, D. R., M. F. Schwartz, et al. (1996). Harvest User's Manual. Boulder, University of Colorado.

Honkela, T., S. Kaski, et al. (1996). Newsgroup Exploration with WEBSOM Method and Browsing Interface. Helsinki University of Technology, Faculty of Information Technology.

Huibers, T. (1996). Towards an axiomatic aboutness theory for information retrieval. 2nd Workshop on Information Retrieval, Uncertainty and Logic, University of Glasgow, Scotland.

Kohonen, T. (1995). Self-organising maps. Berlin, Springer Verlag.

Koster, M. (1995). “Robots in the Web: threat or treat?” ConneXions **9**(4).

Kowalski, G. (1997). Information Retrieval Systems. Theory and Implementation. Boston, Kluwer Academic Publishers.

Ladha, S. (1998). Evaluation of a PC-based LSI Search Engine. Department of Computer Science. Knoxville, University of Tennessee.

Lai, V. S. and R. Mahapatra (1997). The Implementation of Intranets to Support Corporate Distributed Computing Strategy: Some Hong Kong Experiences. Third Americas Conference on Information Systems, Indianapolis, Indiana.

Landauer, T. K., D. Laham, et al. (1997). How well can passage meaning be derived without using word order: A comparison of Latent Semantic Analysis and humans. Proceedings of the Cognitive Science Society.

Lee, L. J. (1997). Similarity-Based Approaches to Natural Language Processing. Computer Science. Cambridge, Massachusetts, Harvard University**:** 116.

Letsche, T. A. and M. W. Berry (1997). “Large-Scale Information Retrieval with Latent Semantic Indexing.” Information Sciences - Applications **100**: 105-137.

Liere, R. and P. Tadepalli (1996). The Use of Active Learning in Text Categorization. AAAI Spring Symposium on Machine Learning in Information Access, Stanford.

Ljungberg, F. and C. Sørensen (1998). Are you pulling the plug or pushing up the daisies? The Thirty-First Hawaii International Conference on System Sciences, Hawaii, IEEE Computer Society Press.

Ljungstrand, P. and H. Fagrell (1998). Make an Agent and You Shall Find. IRIS 21, Denmark.

Luhn, H. P. (1957). “A statistical approach to mechanized encoding and searching of literary information.” IBM Journal **1**(4): 309-317.

Merkl, D. and A. M. Tjoa (1996). Data Mining in Large Free Text Document Archives. International Symposium on Cooperative Database Systems for Advanced Applications (CODAS'96), Kyoto, Japan.

Muñoz, A. (1997). “Compound Key Word Generation from Document Databases Using A Hierarchical Clustering ART Model.” Intelligent Data Analysis **1**(1).

Nelson, M. R. (1994). “We have the information you want, but getting it will cost you: Held hostage by Information Overload.” ACM Crossroads **1**(1).

Oard, D. W. and G. Marchionini (1996). A Conceptual Framework for Text Filtering. College Park, MD, University of Maryland.

Petersson, J. (1993). Tillämpad linjär algebra. Kungsbacka, Sweden, Rex Offsettryck.

Rijsbergen, C. J. v. (1979). Information Retrieval. London, Butterworths.

Salton, G. (1989). Automatic Text Processing. Reading, Addison-Wesley.

Salton, G. and M. Lesk (1965). “The SMART automatic document retrieval system - an illustration.” Communications of the ACM **8**(6): 391-398.

Sommerville, I. (1996). Software Engineering, Addison-Wesley.

Starius, G. (1996). Numerisk analys för E2. Göteborg, Institutionen för datavetenskap CTH/GU.

Wurman, R. S. (1989). Information Anxiety. New York, Doubleday.