INTERACTIVE INFORMATION ORGANIZATION: TECHNIQUES AND EVALUATION

A Dissertation Presented

by

ANTON LEUSKI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May, 2001

Department of Computer Science

INTERACTIVE INFORMATION ORGANIZATION: TECHNIQUES AND EVALUATION

A Dissertation Presented

by

ANTON LEUSKI

Approved as to style and content by:

James Allan, Chair

W. Bruce Croft, Member

David D. Jensen, Member

Christopher Raphael, Member

James F. Kurose, Department Chair
Department of Computer Science

ACKNOWLEDGMENTS

I am deeply grateful to my advisor James Allan for his support and guidance through all my work. To paraphrase a Russian proverb, he helped me to see the forest behind the trees – he helped me to stay aware and appreciate the bigger picture behind the small details of the everyday research problems.

I also thank the rest of my dissertation committee: Bruce Croft, David Jensen, and Christopher Raphael for their comments and suggestions about this writeup.

I would like to thank Andrew Barto and Michael Kositsky for their advice and comments about the reinforcement learning part of the thesis.

Victor Lavrenko helped me to provide the Lighthouse system with access to the Web search engines, taking it from purely research settings into the wild world of the Internet. The intense discussions with Russell Swan were a huge inspiration for this study. My appreciation also goes to Stephen Cronen-Townsend, David Fisher, David Frey, Dawn Lawrie, and Jeremy Pickens who took on the task of checking the parts of the writeup for language and grammar.

This material is based on work supported in part by the Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623, and in part by the National Science Foundation under grant number IRI-9619117, the SPAWARSYSCEN-SD grant number N66001-99-1-8912 and by Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235.

Any opinions, findings and conclusions or recommendations expressed in this material are the author and do not necessarily reflect those of the sponsor.

ABSTRACT

INTERACTIVE INFORMATION ORGANIZATION: TECHNIQUES AND EVALUATION

MAY, 2001

ANTON LEUSKI

M.Sc., MOSCOW STATE UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS AT AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor James Allan

The explosive growth of digital information available on-line and ubiquity of the Internet require development of effective techniques for information search and access. Locating interesting information on the World Wide Web is the main task of on-line search engines. Such an engine accepts a query from a user and responds with a list of documents or web pages that are considered to be relevant to the query. The pages are ranked by their likelihood of being relevant to the user's request. The majority of today's Web search engines follow this scenario.

The ordering of documents in the ranked list is simple and intuitive. The user is expected to follow the list while examining the retrieved documents. In practice, browsing the ranked list is rather tedious and often unproductive. Existing evidence shows that users quite often stop and do not venture beyond the first screen of results or the top ten retrieved documents. In this thesis we study alternative document organization techniques which can help the user to find the relevant information in the retrieved data much more quickly than the ranked list.

The performance of an interactive system is determined by both the system components and the user of the system. We introduce a novel evaluation approach that is based in part on modeling the system-user interaction. It allows us to separate the user's effect on the overall performance from the system qualities.

We apply this evaluation method to two different document organization techniques. The first technique exhaustively partitions the document set into well-defined groups. The second system visualizes documents as objects in space arranged in proportion to the inter-document similarity. We show that both systems can be used much more effectively than the ranked list approach. We use a reinforcement learning algorithm to build a "wizard" tool that helps the user to navigate the system. This wizard provides better support than traditional relevance feedback. In conclusion we illustrate the application of the technologies evaluated in the thesis on an example of document organization interface system for a web-based search engine.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

It is becoming more evident among the Internet community that on-line searches are very inefficient. More often than not people walk away from a search session frustrated:

> "On average, Web-rage is uncaged after twelve minutes of fruitless searching [using a web-based search engine], although about seven percent of the 566 people surveyed by Roper Starch Worldwide say ire starts rising within three minutes.
>
> "A great majority, (86 percent) of Internet users feel that a more efficient way to search the Web for accurate information should be in place," Roper Starch Worldwide researchers wrote."
>
> Ben Charny, ZDNet News.
> http://www.zdnet.com/zdnn/stories/news/0,4586,2667216,00.html
> December 23, 2000

The same study reflects that the main culprit of the "web-rage" is the overwhelming amount of material returned by the search engines. Finding interesting information among these documents is proving to be a frustrating and difficult task for a casual user.

This thesis looks at the problem of locating relevant information among the documents returned by an information retrieval system. We believe that an important key to solving this problem and ultimately improving the search process is an effective organization and presentation of the retrieved material.

Generally a search engine presents the retrieved document set as a simple list of document titles. The documents in the list are ranked by the probability of being relevant to the user's request. The highest ranked document is considered to be the most likely relevant document, the next one is slightly less likely and so on. This organizational approach, called a *ranked list*, is widely popular and can be found in almost any existing search engine [45, 54, 42]. It is assumed that the user will start at the top of the list and follow it down examining the documents one at a time. An ideal information retrieval system will retrieve all and only relevant documents. Or, considering the retrieval as imposing an order on the collection documents, it will rank all the relevant documents in the collection above any non-relevant document. In this thesis we describe and analyze two alternative information organization techniques. We experimentally show that both are more effective than the ranked list.

We continue this chapter with an example that illustrates how grouping similar documents together can help the user to find the relevant material more quickly than a simple ranked list. We then define document organization as a human-centered and highly interactive process with two major components: (1) arranging the documents into an effective structure and (2) presenting that structure to the user. We outline the thesis and state our contributions.

1.1 An Example

Imagine a student who is doing a research project on the movie "Star Wars" and is interested in all the information related to the movie and the entertainment business. She has the archives of the Wall Street Journal from 1987 available and she decides to search them. The articles exist in an electronic format and can be searched automatically by an information retrieval system. Information

Figure 1.1. Top ten retrieved documents for the "Star Wars" query. Gray background indicates relevant documents.

Retrieval (IR) is the area of Computer Science that is concerned with automatic methods for finding relevant information in large collections of documents. Generally such a system accepts a natural language request, decides on the value of every document in the collection, and responds with a list of documents in the order it considers them to be relevant.

Our student turns to the information retrieval system and types in "Star Wars." The system returns fifty documents and shows them on the screen as a ranked list of titles. The screen space is very limited and the returned documents are shown in groups of a few documents at a time. Figure 1.1 shows the first ten documents the student sees on the screen.

Looking at these ten documents more closely, we notice quite a few documents unrelated to the movie; they discuss the Strategic Defense Initiative (SDI) that was nicknamed "Star Wars." These documents are ranked 01, 02, 03, 04, 07, and 09. The document ranked 05 is a long interview with George Lucas and is highly relevant to the student's need. The document 08 might have been relevant as its title talks about television and films, but the document is about a famous movie star participating in a new war movie. The last document is about a wave of Star Wars related rides in theme parks, which the student considers relevant. The search was not very successful since we have two relevant documents among the first ten retrieved. However, there are more relevant documents down the ranked list; recall that we received fifty documents, but we only looked at the first ten.

What should the student do now?

She might continue looking for the rest of the interesting material examining one document at a time – a tedious and time consuming process. The student could abandon these results and try to adjust the query to make it more specific hoping that the new query will bring the relevant documents up in the ranked list. The adjustment can be done by hand, e.g., by typing in "Lucas movie Star Wars." This requires the user to have some basic understanding of why the search process was ineffective. It also forces the user to carefully select the additional terms for the query – the term must not be too general or too specific to focus the query on the relevant documents. Alternatively, the user can mark the relevant documents and apply an automatic relevance feedback tool that will modify the query for her. Such a tool is often supplied with an IR system. While the relevance feedback approach has been shown to produce significantly more effective queries and dramatically improve the search results, it is rather complex, computationally expensive, and difficult to explain. It is failing to attract appropriate usage in casual searches [91].

Despite all the effort – by the user or by the automatic relevance feedback tool – there is no guarantee that the new query will find any more relevant information than we already have in our ranked list. There are more relevant documents in the list, they simply are scattered among the non-relevant information. Moreover, if the archive provider charges a fee for each search session, the new search might be wasting the user's both time and money.

In this thesis we suggest an alternative approach – we show how a content-based information organization can help the user locate the interesting documents quickly. Let us continue with our example. Now suppose that the student takes the retrieved documents and turns to an information organization system that groups similar documents together. The system organizes the fifty retrieved documents into ten groups or clusters. Figure 1.2 shows the highest ranked document for each cluster.

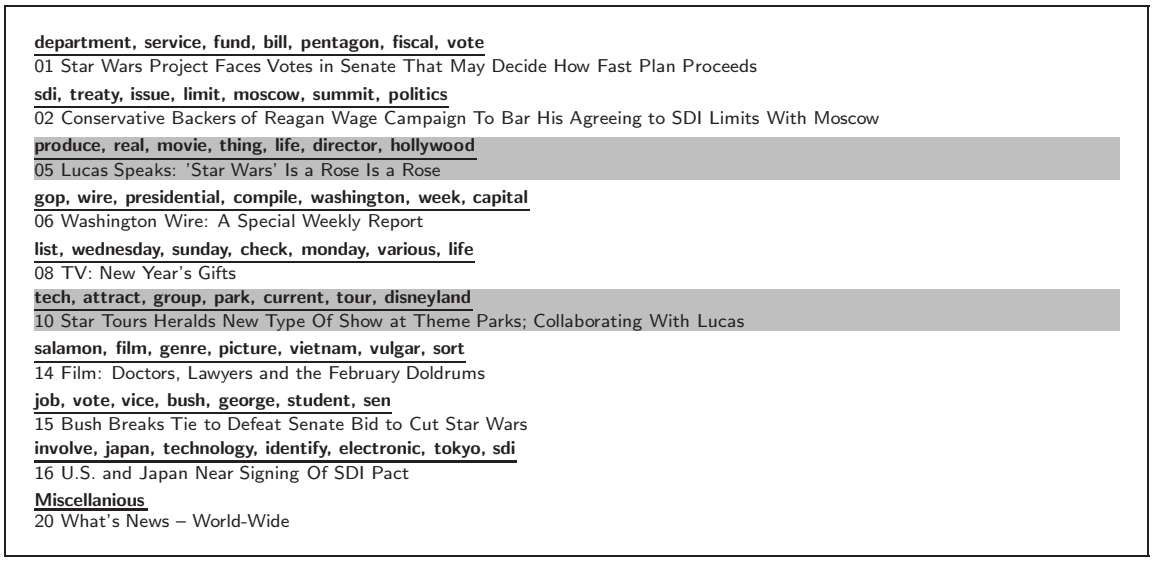

Figure 1.2. A set of clusters created for the top fifty documents returned for the "Star Wars" query. The highest ranked document from each cluster is shown. Also we show seven the most "important" terms for each cluster. Gray background indicates relevant documents.

Figure 1.3 presents the complete structure. The clusters are ordered using the rank of the highest ranked document in each cluster. By looking at the first document in each cluster the student can easily dismiss both the first cluster (it is about SDI and senate) and the second cluster (it is about SDI and Russia). She arrives at the third cluster and finds its first document highly relevant. She examines the rest of the documents in the cluster – they are also relevant. Note that the documents in this cluster were ranked 05, 45 and 50. If the student had continued traversing the ranked list, she would have spent a lot of time searching before finding these documents.

Now the student marks the first and the second clusters as non-relevant and the third as relevant – she provides the system with information on 18 documents after examining only 5 of them. The system uses this information to recluster the documents and brings together the third and the sixth clusters. The documents ranked 10, 13, and 28 are about the movie's impact on the theme park industry. The student also considers them relevant. The clustering proves very helpful as it allows the student to make the relevance judgments on the whole cluster after examining only one document from it – the documents that are similar to a relevant document are very likely to be relevant as well.

Note that by following this strategy the user looks at 8 documents and finds all relevant information in the top portion of the ranked list. By following the ranked list, she would have to look at all fifty documents to get to all relevant documents.

That example illustrates the problem we address in this thesis. Generally an IR system is very successful in finding a set of relevant information in a large collection of documents. However, its success can be hindered by inadequate representation of the results, i.e., when the relevant documents are scattered across the retrieved set and fail to appear at the top of the ranked list. This failure could be attributed to many different reasons. For example, the user's information need is vague and unshaped, or the user has difficulties in expressing her need in a sufficiently clear and unambiguous form. Sometimes ambiguity arises inadvertently due to the particular content of the document collection or the user's erroneous expectations of that content (as in our example). The cause of this problem matters little, as the result remains – the user has to spend a lot of effort isolating the relevant material among the returned documents. In the discussion that follows we will show how alternative information organization techniques help the user to find the interesting information quickly.

department, service, fund, bill, pentagon, fiscal, vote
01 Star Wars Project Faces Votes in Senate That May Decide How Fast Plan Proceeds
03 Weinberger Says Star Wars Would Aid, Not Replace, Nuclear Weapons Arsenal
04 House Allows $3.1 Billion for Star Wars In Fiscal '88, Far Shy of Reagan's Wish
11 Senate Passes Measure to Curb Awarding Of Star Wars Contracts to Foreign Firms
26 Pentagon Seeks Vehicle to Test Space Weapons
44 Senate Votes to Cut Star Wars Funding, Hasten Search for Nuclear Waste Dump

sdi, treaty, issue, limit, moscow, summit, politics
02 Conservative Backers of Reagan Wage Campaign To Bar His Agreeing to SDI Limits With Moscow
07 Missile Fight: Reagan's Arms Policies Are Damaged by Acts Of His Warring Aides
09 Washington Insight: Gorbachev Laid On PR Feast for U.S., But Summit Leaves an Empty Feeling
12 Politics Push U.S., Soviets Into Summits
17 SDI Could Lead a Reagan Revival
18 Gorbachev Cites Possibility of 50% Cut In Superpowers' Long-Range Missiles
22 Moscow's Bigger Star Wars Drive
25 Reagan Foresees Accord to Curb Strategic Arms
42 Letters to the Editor: Is There a Star Wars Gap?

produce, real, movie, thing, life, director, hollywood
05 Lucas Speaks: 'Star Wars' Is a Rose Is a Rose
45 Large Number of Holiday Films May Give Hollywood a Record Year at Box Office
50 Film: Laughs in Space From Brooks and Spielberg

gop, wire, presidential, compile, washington, week, capital
06 Washington Wire: A Special Weekly Report
19 Washington Wire: A Special Weekly Report
23 Washington Wire: A Special Weekly Report
39 Washington Wire: A Special Weekly Report

list, wednesday, sunday, check, monday, various, life
08 TV: New Year's Gifts
27 Television Preview: Opera, Oscars and Music Makers
29 Television: Fox Has a Star in 'Our Trace'
38 TV Preview, Feb. 19-March 4: Aerobics From Russia
43 TV: Gearing Up for the Peoplemeter
48 Television: Politicians and Protestants

tech, attract, group, park, current, tour, disneyland
10 Star Tours Heralds New Type Of Show at Theme Parks; Collaborating With Lucas
13 Creating Fantasies: Disney's 'Imagineers' Build Space Attraction Using High-Tech Gear
28 Joy Rides: Theme Parks Introduce More High-Tech Thrills and Chills

salamon, film, genre, picture, vietnam, vulgar, sort
14 Film: Doctors, Lawyers and the February Doldrums
30 Film: Babes in Filmland: A Fete for Student Directors
34 On Film: The Vietnam War at Arm's Length

job, vote, vice, bush, george, student, sen
15 Bush Breaks Tie to Defeat Senate Bid to Cut Star Wars
40 Bush, Blue-Blood, Texas Oil Man, Washingtonian

involve, japan, technology, identify, electronic, tokyo, sdi
16 U.S. and Japan Near Signing Of SDI Pact
32 Violations of the Technology Export Law
33 Tokyo Rejects the Accusation

Miscellanious
20 What's News – World-Wide
21 TECHNOLOGY: Superconductor 'Battery' Gets A Boost From Star Wars Role
24 Bookshelf: A Bold Rx for Pentagon Ills
31 Politics '88 (A Special Report) — The Candidates
35 Defects Have Killed Sailors, Maimed Sick, Hurt Firms; — Is 'Star Wars' Too Vulnerable?
36 Star Wars' First Phase Would Include Space-Based Arms, Missiles on Ground
37 A Politician's Past Isn't a Drawback In Italy's Parliament
41 The Reagan Legacy – Reagan's World: Foreign Stance of U.S.
49 Kemp Joins GOP Race for Presidency, Vows to Make It Star Wars Referendum

Figure 1.3. A set of clusters created for the top fifty documents returned for the "Star Wars" query. Gray background indicates relevant documents.

1.2 Information Organization

How does Information Organization relate to Information Retrieval? Both these disciplines deal with helping the user to *find* the relevant information. However, an information retrieval system is an active system. It takes a query, searches a large collection of document, retrieves documents that are similar to the user's request, and imposes a ranking order on the returned documents. The user is expected to follow the system recommendations – she is expected to view the documents in the order returned by the system. In contrast, we consider an information organization system as a more passive player. The system uses its knowledge about the document content, origin, or any other available attributes, it structures the documents, visualizes the material, and provides clues to the document content, but the actual "retrieval" is done by the user. The user decides which documents to view and in what order.

Ideally a search system should combine both approaches: information retrieval to isolate a subset of the document collection that is similar to the user's query and information organization to help the user understand what has been retrieved and help her to locate interesting information among the retrieved data. Thus, we place information organization as a post-process in the information retrieval scenario.

For this thesis we define information organization as introducing a content-specific structure into a set of documents and presenting the structure to the user. The documents are retrieved by an information retrieval system in response to a user's query. We assume that these documents contain text as opposed to images or sounds. Building and visualizing the document structure is done for the purpose of helping the user to assess the content of the document set and locate relevant documents among the retrieved material.

Figure 1.4 gives a few examples of what we consider information organization. There we see an example of the clustering approach we presented at the beginning of this chapter. The second example shows documents organized into a tree and presented as a file-folder hierarchy. The third example presents the documents as circles on a plane, where the distance between a pair circles is proportional to the similarity between the corresponding documents. Two groups (or clumps) of similar documents are easily recognizable.

There are three major aspects in our analysis of an information organization system. The first two aspects are about designing the system and the last one is evaluating it.

To design the system we first have to consider how we are going to organize the information and define the structure into which we arrange the information. In our examples the documents are grouped into clusters based on their content similarity. The system computes the inter-document similarities and makes the decision of which documents are to be considered similar enough to be brought together in the same cluster.

The next step is to present the organizational structure to the user. We see the clusters as a list of document titles and the document titles from each cluster are shown together. Note that both the structural organization and presentation are tightly connected as the presentation is generally designed to reflect the structure and convey it to the user. At the same time they are separate from each other as different presentations can be used to visualize the same organizational structure.

The last aspect of this study is the evaluation of an interactive information organization system. We view information organization as a post-process in the information retrieval scenario. Measuring the ability of the system to help the user locate the interesting information is the main test of the design validity. We consider the evaluation an inherent part of our study.

1.3 Thesis Outline

In this thesis we analyze two interactive information organization systems. The first system employs a well-known clustering algorithm to partition the retrieved set into the groups of similar documents. The second system applies a multidimensional scaling algorithm called *spring-embedding* to represent the documents as objects in 2 or 3 dimensions positioned in proportion to the inter-document similarity. Both systems use the same information – the inter-document similarity – to organize the search results in two very different ways.

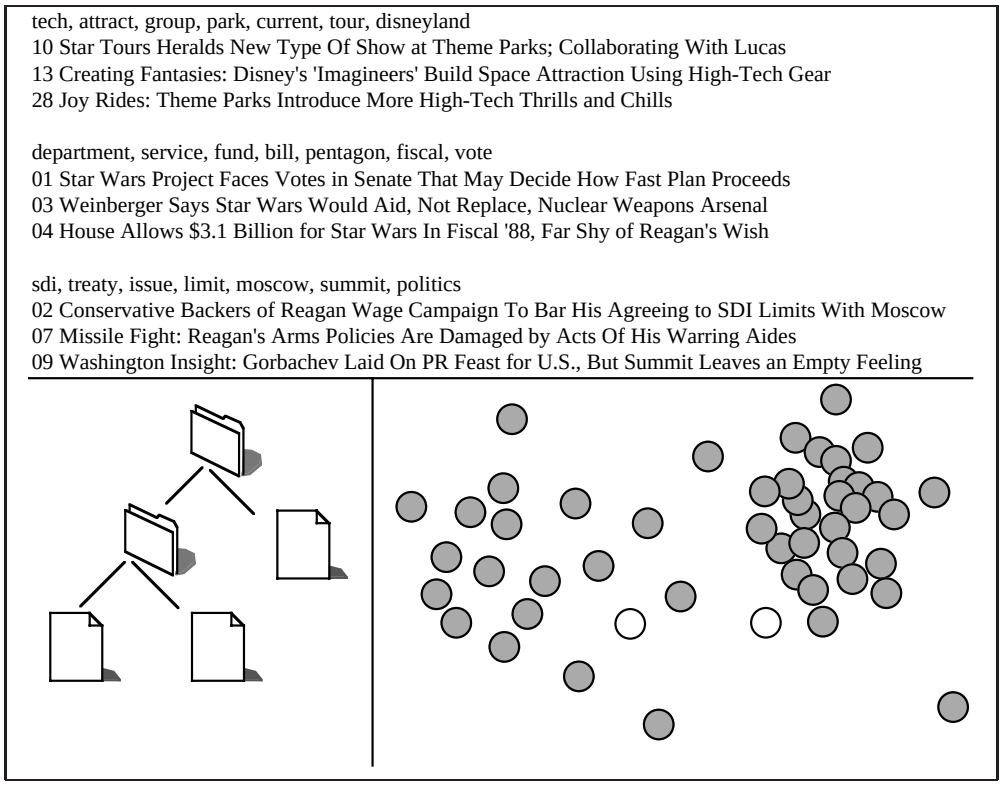

Figure 1.4. Three examples of information organization

In Chapter 2 we establish our evaluation framework. The evaluation is task-oriented and for this study the task is specified as finding the relevant documents in the retrieved set. The evaluation framework proceeds by replacing a real user with an algorithmic simulation of a search process and we define two different simulation strategies. These strategies are applied to each system and executed on a standard set of collections, queries, and relevance judgments (i.e., data from the *Text REtrieval Conference (TREC)* [46]). We call this approach the *Strategy-Based Evaluation* (SBE) method. We also describe the evaluation measure, introduce the ranked list and interactive relevance feedback as our baseline systems, and define their performance on the dataset. While both systems in our study are well-known, the analysis methodology is novel.

The Strategy-Based Evaluation framework is applied to evaluate an information organization based on an agglomerative clustering algorithm in Chapter 3. We have briefly introduced the system in Section 1.1. We describe the system design and consider several questions including comparison of different clustering algorithms and how the clustering hierarchy is converted into a partition of the document set. Our analysis shows that a user can find relevant information among clusters more quickly than by traversing the ranked list. However the effectiveness of the clustering organization falls short of the performance displayed by the interactive relevance feedback – our second baseline system. Our extensive analysis of different clustering algorithms and strong experimental results obtained in that chapter are important contributions of the thesis.

Partitioning the document set into non-overlapping clusters simplifies the inter-document relationships. It is very easy from a user's point of view to tell a pair of similar documents from a pair of dissimilar ones – similar documents are assigned to the same cluster. This simplification comes at a cost of losing the intricate details of the inter-document similarity, that might otherwise be useful for locating relevant information. Additionally, the clustering algorithm is a parametric approach. Determining the best value for the parameter that defines the final partition of the document set is a very hard question which we would like to avoid in a real-world system.

We address the limitations of the clustering approach in Chapter 4. That chapter presents an information organization system that is based on the spring-embedding visualization algorithm. We apply the Strategy-Based Evaluation framework to show that the visualization contains sufficient information to make browsing the spring-embedder-created picture much more effective than using the ranked list and as effective as using the relevance feedback. We also show that the dimensionality of the visualization does not significantly affect the retrieval quality of the visualization. A user study reveals that users do not recognize and use all the available information in the visualization – average user's performance, while significantly exceeding that of the ranked list, falls short of the relevance feedback baseline.

These results suggest that while the spring-embedding visualization is powerful enough for one to expect great performance, casual users may require some assistance which the information organization system can provide. For example, this help can come in the form of pointing out which documents are the most similar to the ones considered. In Chapter 5 we formulate the task of locating relevant documents as a reinforcement learning problem and show how a modest amount of training helps us to develop an effective assistance tool for an information organization system that uses inter-document similarity information.

Chapter 6 brings the results of this thesis together and shows how they can find a practical application in the web-based search environment. We describe Lighthouse, an interface system for an on-line search engine that we have developed and implemented for this thesis. It combines the ranked list, clustering approach, and spring-embedding visualization. It also incorporates the search assistant developed with the reinforcement learning algorithm.

We summarize the main results of this thesis in Chapter 7 and discuss directions for future work.

1.4 Contributions

This thesis makes the following broad contributions to the field of information retrieval:

- We study two information organization techniques: (1) document clustering and (2) spring-embedding visualization. These approaches to document organization are well-known, however, we are the first to study them in the context of ranked document retrieval task. We apply information organization to a set of documents found by an information retrieval system in response to a predefined query. We investigate how the organization helps the user to isolate interesting material in the retrieved set quickly. Also the extent of our analysis and the questions considered are the prominent features of this work.

- We investigate a method for evaluating interactive information organization systems that can be carried out in the absence of a user. We study applications of this method to measuring the retrieval effectiveness of information organization systems – how well the systems support a user in finding interesting information. This approach is significantly different from what has been attempted before.

Specifically, we contribute the following:

- We define an interaction model between an information organization system and a user. We design a novel method for evaluating interactive information organizations that does not require immediate user participation – we simulate the user with an artificially constructed (though plausible) user strategy.

- We compare six different hierarchical agglomerative clustering algorithms. We define an effective method for transforming a clustering hierarchy into a partition of the document set. We show that these document set partitions can be much more helpful in locating the relevant information than the traditional ranked list.

- We study a visualization system that places documents in 2- and 3-dimensional space. We are the first to investigate how successful the visualization is in helping a user to locate interesting information in the retrieved document set. We show that such a visualization can be more beneficial than the traditional ranked list approach for presenting the retrieval results.

- We show that the spring-embedding visualization of inter-document similarities is accurate enough for the retrieval task. Using the proximity information between objects in the 2- and 3-dimensional projections of the document set to isolate relevant documents is as effective as using the actual inter-document similarities for the same task.
- We show that while a 3-dimensional visualization is more accurate than a 2-dimensional one, the users are more effective and comfortable with the 2-dimensional interface.
- We state the task of locating the relevant information in terms of reinforcement learning. We define an automatic agent that performs this task on a retrieved document set and show that the browsing effectiveness of such an agent is significantly improved after a modest amount of training.
- We present a system that integrates the traditional ranked list with document clustering and spring-embedding visualization of the retrieved documents. The system supports the user's browsing process with a wizard tool constructed using the reinforcement learning approach. The system's design builds on the results from the work in this thesis.

CHAPTER 2

EVALUATION METHODOLOGY

The focus of this chapter is our evaluation approach. We begin by defining a model of a user interacting with an information organization system and construct the evaluation framework around that model. The evaluation consists of two steps: (1) isolate the information supplied by the document organization and show that it can be used to improve search effectiveness and (2) show that users can recognize and apply that information. For example, in this thesis we study a spring-embedding visualization that arranges the documents in space in proportion to their similarity. We will show that (1) the spatial distances between document representations can be used to speed up the search process and (2) users are able to recognize and correctly interpret the distances.

We introduce the Strategy-Based Evaluation framework for the first step in our analysis. The framework is based on the interaction model and works by simulating a user with an algorithmic search strategy. The second part of the analysis requires a user study. We also describe the experimental setup and present two control systems used in this analysis – the ranked list and interactive relevance feedback.

We conclude the chapter with an overview of the related work on evaluation of interactive search systems, modeling the information access process and the user's role in that process.

2.1 Interaction Process

Recall our example from Chapter 1 where the student was looking for the "Star Wars" documents. Figure 2.1 depicts the relationship between the user and the system. Here the user has a goal to achieve, a *task* to complete (e.g., find the relevant documents about the movie). The user turns to an information organization system that analyzes the provided information (the retrieved documents), builds a *data model*, and uses the model to organize the data. The organization (e.g., the clustering structure) is shown to the user along with some additional aspects of the model that are not explicitly represented in the organizational structure (e.g., document titles and cluster descriptions). The information reaches the user in the form of *clues*: similarity is characterized by cluster membership or proximity; document content is represented by its title or a small set of highly ranked terms. It is for the user to recognize and interpret the supplied clues as effectively as possible, then to apply this knowledge and decide what documents to view and in what order. We call this decision process the *search strategy*. We saw an example of this process when the student was looking at one document from each cluster and deciding to accept or reject the cluster based on the relevance value of that document. The strategy might be as simple as selecting the next document randomly, or something more elaborate that progresses by taking into account the content of the documents that are being considered and their position in the organizational structure (e.g., cluster co-location with known relevant documents). After a document is selected, the user makes a relevance assessment and passes the judgment to the system. The system takes the *user's feedback* and adjusts the data model (e.g., reclusters the documents). This interactive process continues until the user chooses to stop examining the documents.

The described interaction model defines a general process in which the system and the user are working together to complete a predefined task. Both serve as two components of a large "engine" that drives towards that goal. We are interested in measuring the system's effect on the overall quality of the engine – we want to measure how well the system is able to support a random user in completing the task.

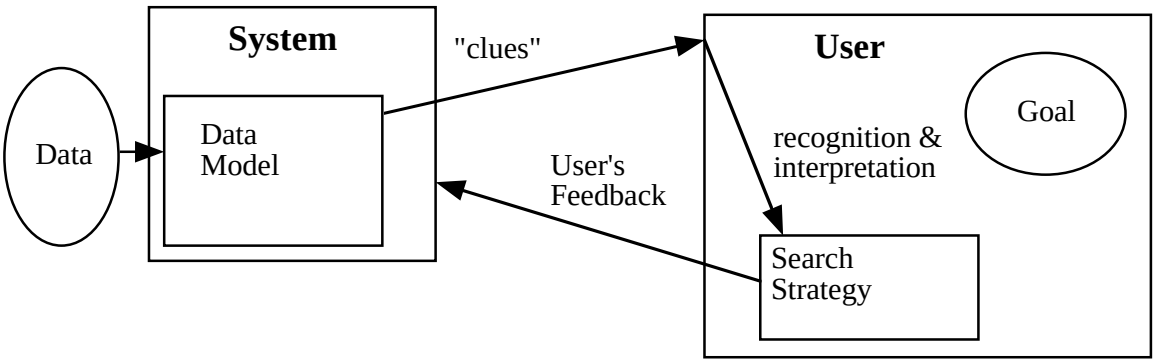

Figure 2.1. The relationship between the system and the user in an interactive information organization setting.

2.2 Informational Clues

It is important for our analysis to note that the user examines the documents sequentially. Thus the user's search strategy results in a document ordering. That ordering is likely to be different from the ranked list as returned by the retrieval system. Nevertheless it is an ordering and we can compare two different information organization systems by comparing the orderings that results from the user's interaction.

What is it that makes two different information organization systems produce two different document orderings? If we assume that the document set is the same and the same user is interacting with both systems, then it is the information organization and presentation of the documents that make the difference. In terms of our interaction model it is the system clues "transmitted" to the user that matter. Thus we focus our analysis on those clues.

The system clues have to satisfy two conditions: first, they have to carry useful information that can improve the effectiveness of the user-system performances. Second, they have to be clearly understandable and interpretable by the user, otherwise the informational benefit will be lost in transmission. Thus we organize our analysis by considering both the utility of the clues and their understandability.

Note that there is a third aspect to the informational clues – the usability aspect. For example, the same spatial configuration of spheres representing the retrieved document set can be perceived differently depending on the color of the spheres, their relative size, contrast, and view angle. The choice of the interface control actions for manipulating the structure can significantly affect the user's interaction as well. Making an incorrect decision at that stage can potentially ruin all the advantages provided by the information organization system. However, we do not consider the questions of usability in our work. Our goal is to show that there exists a measurable objective benefit in using these information organization systems. Accomplishing this objective justifies a further study of the systems with a focus towards usability issues.

To conduct the first part of the analysis and measure the usefulness of clues we replace the user with a probabilistic model of the search strategy. Then, we apply the system and the artificial search strategy to a known data set and measure how well it handles the task. We call this approach the *Strategy-Based Evaluation* (SBE) method. The quality of the system becomes a function of both the task and the search strategy. In every case we specify the components of the interactive model. These include the *experimental task*, *system design*, *search strategy*, *performance measure*, and *experimental setup*. The parts that cannot be specified exactly have to be approximated. Then the interaction process is simulated on an experimental data set. The performance measure provides an estimate of the system quality.

Specifically, we define a simple algorithmic search strategy and adapt it both to the clustering and the spring-embedding systems. With each system the search strategy uses the system's corresponding

information clues and orders the documents simulating a user working with the system. We evaluate the performance of the search strategy and compare it with our baseline data. We show that the clues provided by the information organization systems can be used to improve the effectiveness of the search process.

Note that such an analysis results in a lower bound estimate of the system performance. It is possible that a user can find a more effective way of applying the informational clues supplied by the system. On the other hand, if the user adheres to the choices made by the algorithmic search strategy, she is guaranteed at least the level of performance predicted by the SBE analysis.

The second part of the analysis is to test whether the users can understand and interpret the clues supplied by the system. We believe some of the clues can be assumed to be understood perfectly. For example, an ordered set of documents indicates that these documents should be examined sequentially starting at the top of the list and following it down. We assume that this is clearly understood and easily interpreted by any user. We also assume that presenting two documents in a group can be clearly recognized as these documents being similar to each other and placing two documents in separate groups indicates unrelated document content. Thus we assume that the clues used in the clustering system can be interpreted correctly by any user.

The ability of a user to recognize and interpret the spatial proximity clues between objects correctly is a more complex question. In Section 4.2 we describe a specialized user study that we conducted to test this question. That study focuses on whether the users can view the spatial proximity as a metaphor for similarity between objects and whether they can locate the target objects by navigating the spring-embedded visualization.

2.3 Strategy-Based Evaluation

This section defines the main components of the Strategy-Based Evaluation framework common to all the systems in our study. The unique details of each system, specifically parts of the *system design* and *search strategy* are defined in the corresponding chapters.

2.3.1 Experimental Task

Our evaluation approach is task-oriented – we assume that the user is working with the information organization system to find the relevant material among documents retrieved by an information retrieval system. In the experiments that follow we consider two different initial conditions for this search process. With the first alternative we assume that the user and the system are working on the document set "as-is", i.e., no relevance judgments about the documents are available to the user at the beginning of the process. We call this initial condition the *no-information* condition. The experimental task with the no-information condition is defined: *Given that no documents presented by the information organization system are marked as relevant or non-relevant, isolate the relevant material in the document set.*

For the other setup we assume that the user has studied some of the documents and is aware of their relevance value. Now she wants to find the rest of the relevant documents as quickly as possible. Specifically, we simulate a situation when the user starts reading documents from the top of the ranked list and continues until she finds one relevant document. This gives a good example of the user providing the system with minimum relevant information (one relevant document). Thus we consider the ranked list and consider the highest ranked relevant document (assuming that it exists). We assume that we know the relevance judgments for this document and for all documents that precede it in the ranked list (of course they all are non-relevant). We call this initial condition the *first-relevant* condition and we call the set of documents that are assumed to be examined *first-relevant document subset*. Thus, the experimental task with the first-relevant initial condition is defined: *Given that some of the documents presented by the information organization system are marked as relevant or non-relevant, isolate the rest of the relevant material.*

2.3.2 System Design

The system design part of the framework describes how each system in our study builds its data model. We specify what kind of feedback the system accepts and how this information affects the data model. The interface or the visualization part of the system is also very important. For each system we describe what information about the data model is communicated to the user and how these clues are presented.

Both the ranked list and interactive relevance feedback document organization systems provide the baseline for our experiments. The ranked list is generated by the INQUERY system [7]. INQUERY uses an inference network model and estimates probabilities of how much each document satisfies user's information need [99]. The intermediate nodes in the network correspond to semantic concepts and the leaves represent the individual terms. An ad-hoc term weighting formula [7], which combines Okapi's *tf* score [80] and INQUERY's normalized *idf* score, is used to estimate conditional probabilities of a document containing a semantic concept given term:

$$w_{i,j} = \frac{tf_{i,j}}{tf_{i,j} + 0.5 + 1.5\frac{doclen}{avgdoclen}} \cdot \frac{\log(\frac{colsize+0.5}{docf_i})}{\log(colsize+1)} \tag{2.1}$$

where $w_{i,j}$ is the weight of the i-th term in the vocabulary in the j-th document, $tf_{i,j}$ is the number of times the term occurs in the document, $docf_i$ is the number of documents the term occurs in, *doclen* is the number of terms in the document, *avgdoclen* is the average number of terms per document in the collection, and *colsize* is the number of documents in the collection.

The INQUERY's retrieval model neither incorporates the notion of similarity between documents nor assumes the construction of document representations. To compute inter-document similarities we employ the vector-space model for document representation [85] – each document is defined as vector V, where $v_i = w_i$ is the weight in this document of the i-th term in the vocabulary. The term weight is computed following the INQUERY weighting formula. The similarity between a pair of documents is computed as the cosine of the angle between the corresponding vectors ($\cos\theta$) [85]. In this thesis we use one over the cosine ($1/\cos\theta$) to define the dissimilarity (or the *distance*) between a pair of documents.

For our second baseline we selected the interactive relevance feedback approach. Relevance feedback has been a major research focus of information retrieval for well over 30 years and has been shown to dramatically improve retrieval performance [84, 86]. The general model is to examine a portion of the retrieved documents assigning relevance values to each of them. The document's content is analyzed to "shift" the query closer towards the examined relevant documents and away from the examined non-relevant documents. This process is usually done in batches – the documents in the collection are broken into training and testing sub-collections, the queries are modified with the information from the training collection and evaluated on the test collection.

Aalbersberg [1] explored the notion of incremental relevance feedback, where the documents are considered one at a time and the query is modified after each document. Allan [4] conducted an intensive study of the incremental approach in which the relevant judgments are submitted to the system a few at a time. His experiments showed results as good as if the feedback occurred in one pass.

The most common model for implementing the relevance feedback was outlined by Rocchio [81]. The document terms are ranked based on the weighted sum of the terms weights in the old query, the known relevant, and known non-relevant documents. The new query is constructed with several top ranked terms:

$$w_i(\mathcal{Q}') = \alpha \cdot w_i(\mathcal{Q}) + \beta \cdot \overline{w_i(\mathcal{R})} + \gamma \cdot \overline{w_i(\mathcal{N})} \tag{2.2}$$

here $\overline{w_i(\mathcal{R})}$ and $\overline{w_i(\mathcal{N})}$ are the average weights of the ith term in the relevant and non-relevant documents, $w_i(\mathcal{Q})$ and $w_i(\mathcal{Q}')$ are the weights of the same term in the old and new queries. Parameters α, β, and γ – called *Rocchio coefficients* – control the relative impact of each component. Generally, these parameters are selected empirically and the best 10-20 terms are added to the query [7]. Buckley and Salton [20] suggested an approach called Dynamic Feedback Optimization (DFO) where the

coefficients are learned by greedy exploration of the parameter space. They also demonstrated that increasing the number of expansion terms can improve performance.

We define the interactive relevance feedback procedure as follows: we start from the top of the ranked list. Each time a new relevant document is discovered, we submit all the examined documents to the INQUERY's relevance feedback subsystem to modify the weights in the original query. Additionally, the query is expanded by adding several highest ranked terms from the examined documents [4]. Note that this procedure takes into account both relevant and non-relevant documents. The unexamined documents in the set are reranked by INQUERY using the new query and we continue down the list.

2.3.3 Search Strategy

Bookstein [18] argues that information retrieval should be envisioned as a process, in which the user is examining the retrieved documents in sequence and the system can and should gather the feedback to adjust the retrieval. We adopt a similar notion while looking at organizing the retrieval results. Recall from Chapter 1 that we consider the organization as a process that induces an order on the retrieved documents and the users are expected to follow that order while examining the results. That process may be "static" – the ordering happens once per retrieval, or it may be "dynamic" – the documents are reordered as the user's feedback is gathered.

For example, to build a ranked list the documents are ordered by probability of being relevant, the user is expected to start at the top of the ranked list and proceed down the list examining the documents one-by-one. We call this expected user's behavior the *ranked list search strategy*. This is an example of a *static* search strategy.

The interactive relevance feedback approach defines another document ordering: the documents are ordered by probability of being relevant, the user is supposed to start from the top and examine the documents until the first relevant document is found. That document is used to modify the query, the unexamined documents are reordered by probability of being relevant to the new query, and the process continues. This is a *dynamic* search strategy and we call this expected user's behavior the *relevance feedback search strategy*.

Ultimately the search strategy defines an ordering of the documents or ranking. However, the word *ranking* is closely associated with the original ranked list. To avoid confusion we call both this type of process and the resulting order in which documents are supposed to be examined the *search strategy*. The search strategy for clustering could be something like: "Select the best cluster. Pick a document from that cluster and examine it. If the document is relevant, examine the rest of the cluster, otherwise pick another cluster."

We formalize the notion of a search strategy by introducing a *search strategy function*. A search strategy reorders the documents in the retrieved set at discrete time steps, i.e., when the user examines a document and provides feedback[1]. The reordering is performed given some representation of the current state of the document set $\mathcal{D}_t$, where t is the time step. A mapping is computed between each unexamined document d and a numeric value: $F(\mathcal{D}_t, d)$. The documents are ordered using these numeric values $F(\mathcal{D}_t, d)$. We call the mapping function the *search strategy function*.

The idea of the search strategy function is very similar to the concept of the document ranking function that constitutes the essence of most search engines. The main difference is that the search strategy function is "dynamic": the value $F(\mathcal{D}_t, d)$, and therefore the current ordering of documents, may depend on what documents were examined and what relevance judgments were assigned by the time t.

Note that a strategy function $F(\mathcal{D}_t, d)$ is not necessarily deterministic; it could be random as long as we have a probability for each part of the strategy determined *a priori*. A probabilistic *recognition value* can be attached to each informational clue provided by the system. Then the user strategy is defined as a decision making process. For example, a document summary has a probability associated with it that user looking just at the summary will correctly recognize a relevant document. Another probability will define how likely the user correctly determines that the document is non-relevant.

[1]Note that only a dynamic search strategy reorders the documents after feedback. An ordering produced by a static search strategy remains constant

2.3.4 Performance Measure

The retrieval process is inherently linear. To make a relevance judgment on a document the user has to read the document first. Because people do not read several documents at the same time, retrieval is a decision process on the order in which the documents are examined. This order could be considered as a new ranking imposed on the retrieved documents. (Remember that the retrieval system ranks the retrieved documents according to their similarity to the query.) It is the quality of this new ranking that determines the performance of the information organization system. That quality could be assessed in many different ways. In this thesis we use the well-known measure precision: i.e., the proportion of the examined documents that are relevant.

Suppose we are considering an ordering of documents $d_1, d_2, d_3, \ldots, d_n$. For every relevant document d_i in the ordering we define $\mathcal{R}_i$ and $\mathcal{N}_i$, the sets of relevant and non-relevant documents in the ordering from the beginning up to d_i:

$$\mathcal{R}_i = \{d_j : d_j \text{ is relevant and } j \leq i\}$$

$$\mathcal{N}_i = \{d_k : d_k \text{ is non-relevant and } k \leq i\}$$

Then precision P_i for relevant document d_i is defined as

$$\mathcal{P}_i = \frac{|\mathcal{R}_i|}{|\mathcal{R}_i| + |\mathcal{N}_i|}$$

The average of all precision values is known as non-interpolated average precision [45]:

$$P = \frac{1}{|\mathcal{R}|} \sum_{d_i \in \mathcal{R}} P_i$$

where $\mathcal{R}$ is the set of all relevant documents.

This average precision serves as our performance measure. Note that using recall (i.e., the proportion of retrieved relevant documents among all relevant documents $|\mathcal{R}_i|/|\mathcal{R}|$) does not make sense in our setting since we are only interested in the effect among the top 50-100 documents.

Section 2.3.1 defined two different initial conditions on the search process. We also compute two different values of average precision. In our experiments with the first initial condition – no relevant information available – we compute the average precision in traditional way – for all documents in the retrieved set. In the experiments with the second initial condition – the first relevant document is marked at the beginning of the search – we compute the average precision only for the unknown portion of the retrieved set. Thus, we measure the system's ability to find the rest of the relevant material as specified in the experimental task (see Section 2.3.1). Naturally, the document sets that contain fewer than two relevant documents will generate precision of zero.

We apply statistical tests to evaluate the quality of the differences in average precision numbers for the experimental and control systems. Unless otherwise noted we use the paired two-tailed t-test to measure the statistical significance. The cutoff level is set to 5% ($p < 0.05$).

2.4 Experimental Setup

For our experiments we use the data provided by the Text REtrieval Conference (TREC) [45]. Specifically, we use the ad-hoc queries with their corresponding collections and relevance judgments. TREC topics 251-300 are converted into queries and run against the documents in TREC volumes 2 and 4. In the discussion that follows we call these queries TREC-5 queries. Table 2.1 shows the content and statistics of the collections used in our experiments. A TREC topic defines a content-based information need and consists of a short title and a couple paragraphs describing the relevance criterion for the topic. For each TREC topic we consider four types of queries: (1) a query constructed by extensive analysis and expansion [6]; (2) the description field of the topic; (3) the title of the topic; and (4) a query constructed from the title by expanding it using Local Context Analysis (LCA) [108]. Figure 2.2 shows an example of a queries constructed using these approaches.

Table 2.1. The source and the size of the experimental data collections [46].

Volume	Source	Size in MB	Number of Documents
2	Wall Street Journal, 1990-1992	242	74,520
	Associated Press newswire, 1988	237	79,919
	Computer Selects articles	175	56,920
	Federal Register, 1988	209	19,860
4	Financial Times, 1991-1994	564	210,158
	Federal Register, 1994	395	55,630
	Congressional Record, 1993	235	27,922
5	Foreign Broadcast Information Service	470	130,471
	LA Times	475	131,896

Table 2.2. Experimental data set statistics. Two columns show the statistics for the top 50 and 100 retrieved documents for each experimental query set. The first subcolumn in each column contains the number of document sets with more than one relevant document ("# DS"). The second subcolumn shows the average number of relevant documents in those document sets ("$|\mathcal{R}|$"). The third subcolumn contains the average number of non-relevant documents that appear in ranked list before the highest ranked relevant document in those document sets ("$|\mathcal{N}_1|$").

Data set		top 50			top 100		
		# DS	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{N}_1\rvert$	# DS	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{N}_1\rvert$
TREC-5	Full	41.00	14.39	1.78	45.00	20.49	3.76
	Desc	44.00	11.27	3.20	44.00	16.86	3.20
	Title	37.00	11.38	3.43	42.00	15.79	3.36
	ExpTtl	32.00	14.75	3.16	35.00	21.54	3.17
TREC-6	Full	45.00	16.80	1.16	47.00	25.26	2.04
	Desc	47.00	14.00	1.43	48.00	19.77	2.27
	Title	42.00	14.33	1.26	46.00	19.72	3.13
	ExpTtl	44.00	15.52	2.09	46.00	22.35	2.72
total average		41.50	14.06	2.19	44.13	20.22	2.96

In addition, we use TREC topics 301-350 to create queries to be run against TREC volumes 4 and 5. We call this query set TREC-6 queries. Again, the same four different types of queries are constructed, except instead of just using the description field for the second query type, we use both the title and the description field of the topic. (The description fields for topics 301-350 were designed to assume the presence of the title.)

Different queries constructed for the same topic result in different retrieved sets. That allows us to study what effect both the quality of the query and its size have on the information organization.

Our assumption is that during a typical retrieval session a user does not generally look beyond the first screen showing the retrieved material – that is approximately equivalent to ten retrieved documents. Thus, we are interested in analyzing just the top portion of the ranked list. Unless otherwise noted for each query we select the 50 highest ranked documents. Table 2.2 shows the average number of relevant documents in the top portion of the retrieved set for each experimental dataset. The table also shows the average number of non-relevant documents that a user following the ranked list is expected to find before encountering the first relevant document. We compute these values only for document sets that contain more than one relevant document. The table shows how many document sets in each dataset have this property.

full query:

#WSUM (1.0 1.0 #WSUM(1 1 cases 1 medications 1 procedures 1 prior 1 heart 1 surgery. 4 #wsum(1 1 the 2.25 heart 2.25 surgery 0.75 medications 0.75 procedures 3.375 #passage25(#phrase(heart surgery))))

2.0 #filreq(#WSUM(1 1 cases 1 medications 1 procedures 1 prior 1 heart 1 surgery. 4 #wsum(1 1 the 2.25 heart 2.25 surgery 0.75 medications 0.75 procedures 3.375 #passage25(#phrase(heart surgery)))) #uw150(heart surgery medications))

4.0 #WSUM (1.0 1 surgery 0.987143 patient 0.974286 #3 (heart disease) 0.961429 #3 (heart attack) 0.948571 doctor 0.935714 medication 0.922857 hospital 0.91 #3 (bypass surgery) 0.897143 artery 0.884286 #3 (heart surgery) 0.871429 angioplasty 0.858571 dosage 0.845714 #3 (heart bypass surgery) 0.832857 #3 (blood vessel) 0.82 surgeon 0.807143 heartbeat 0.794286 #3 (heart patient) 0.781429 #3 (cost medication) 0.768571 #3 (heart condition) 0.755714 #3 (artery bypass surgery) . . .

description query:

The document will discuss all those cases in which medications and procedures were used instead of or prior to heart surgery.

title query:

Non-invasive procedures for persons with heart ailments

expanded title and description query:

#WSUM (1.0 1.0 #SUM(non invasive procedure person heart ail)

4.0 #WSUM(1.0 1.000000 #3 (treadmill test) 0.987013 catheterization 0.974026 patient 0.961039 artery 0.948052 #3 (heart attack) 0.935065 doctor 0.922078 surgery 0.909091 infection 0.896104 #3 (heart disease) 0.883117 heart 0.870130 aspergillosis 0.857143 ailment 0.844156 disease 0.831169 diagnosis 0.818182 #3 (duke medical center) 0.805195 hospital 0.792208 medicine 0.779221 aids 0.766234 cardiologists 0.753247 procedure 0.740260 treatment 0.727273 cardiolite 0.714286 catheter 0.701299 #3 (blood vessel) 0.688312 #3 (heart ailment) 0.675325 clinicians 0.662338 #3 (heart muscle) 0.649351 illness 0.636364 tuberculosis . . .

Figure 2.2. An example of four different query types constructed for TREC topic 254 *Non-invasive procedures for persons with heart ailments*.

2.5 Baseline Performance

We have measured performance of the ranked list and relevance feedback search strategies on the 50 highest ranked documents for 100 queries of 4 different types. We considered only the document sets with more than one relevant document. We assumed that the system failed on the rest of the queries and the corresponding document sets have precision of zero.

Table 2.3 shows the average precision numbers computed for the first starting condition. We assumed that no relevant information is provided to the search strategies. The first three columns in Table 2.3 show the average precision numbers for three hypothetical search strategies: one places all non-relevant documents before all the relevant, another randomly distributes relevant documents among non-relevant ("random") and the other ranks all relevant documents above all non-relevant ("best"). These numbers provide a scale for the performance results.

The last columns in Table 2.3 show the performance of the interactive relevance feedback and the percent improvement over the actual ranked list created by INQUERY ("original"). The numbers vary significantly if the best 10 ("+10t") or the best 100 ("+100t") terms are added to the query. We have observed that adding more terms (e.g., the best 200 terms) does not improve the performance of the interactive relevance feedback.

The differences indicated are statistically significant except for the datasets retrieved by the long queries (these are the "Full" and "ExpTtl" query sets for both TREC-5 and TREC-6 collections).

Table 2.4 shows the same performance numbers obtained from the ranked list and interactive relevance feedback search strategies for the second starting condition. Here the average precision is computed starting from the first relevant document in the ranked list. That document and all the non-relevant documents that might precede it in the list are considered to be examined by the user and ignored from the computations. The differences indicated are statistically significant except for the datasets retrieved by the long queries on the TREC-6 collection ("Full" and "ExpTtl").

Table 2.3. Performance of the search strategies for the ranked list (RL) and interactive relevance feedback (RF). The first starting condition is in effect. Average precision numbers, percent improvement over the simple ranked list are shown. Besides the actual ranked list quality as generated by INQUERY ("Orig.") we show three hypothetical cases: "Worst" – all the relevant documents placed after all non-relevant, "Rand." – the relevant documents are equally distributed in the list and "Best" – all the relevant are positioned before all non-relevant.

Data set		Ranked List				Relevance Feedback	
		Worst	Rand.	Best	Orig.	+10t (vs RL%)	+100t (vs RL%)
TREC-5	Full	15.40	27.79	82.00	38.49	39.87 (3.58 %)	41.87 (8.77 %)
	Desc	12.52	24.72	88.00	36.06	42.90 (18.98 %)	44.19 (22.57 %)
	Title	11.04	20.94	74.00	33.19	35.86 (8.07 %)	36.29 (9.36 %)
	ExpTtl	12.26	22.11	64.00	29.91	30.98 (3.59 %)	32.64 (9.11 %)
TREC-6	Full	20.84	34.56	90.00	53.67	53.73 (0.10 %)	54.48 (1.50 %)
	Desc	17.11	31.19	94.00	46.66	56.27 (20.60 %)	58.54 (25.46 %)
	Title	16.19	28.40	84.00	46.19	54.55 (18.11 %)	56.23 (21.75 %)
	ExpTtl	18.47	31.68	88.00	51.64	52.63 (1.91 %)	53.12 (2.87 %)
total average		15.48	27.67	83.00	41.98	45.85 (9.23 %)	47.17 (12.38 %)

Table 2.4. Performance of the search strategies for the ranked list (RL) and interactive relevance feedback (RF). The second starting condition is in effect. Average precision numbers, percent improvement over the simple ranked list are shown. Besides the actual ranked list quality as generated by INQUERY ("Orig.") we show three hypothetical cases: "Worst" – all the relevant documents placed after all non-relevant, "Rand." – the relevant documents are equally distributed in the list and "Best" – all the relevant are positioned before all non-relevant.

Data set		Ranked List				Relevance Feedback	
		Worst	Rand.	Best	Orig.	+10t (vs RL%)	+100t (vs RL%)
TREC-5	Full	15.11	27.41	82.00	36.17	39.86 (10.21 %)	44.17 (22.13 %)
	Desc	12.26	24.57	88.00	34.09	45.12 (32.38 %)	50.60 (48.45 %)
	Title	10.90	20.92	74.00	30.95	38.08 (23.05 %)	41.39 (33.73 %)
	ExpTtl	12.29	22.34	64.00	30.95	34.79 (12.38 %)	38.26 (23.61 %)
TREC-6	Full	20.39	33.93	90.00	51.02	53.64 (5.12 %)	54.70 (7.20 %)
	Desc	16.53	30.41	94.00	42.48	56.04 (31.93 %)	60.39 (42.16 %)
	Title	15.66	27.66	84.00	40.10	52.00 (29.66 %)	54.74 (36.49 %)
	ExpTtl	18.32	31.72	88.00	47.21	48.68 (3.10 %)	50.07 (6.05 %)
total average		15.18	27.37	83.00	39.12	46.03 (17.65 %)	49.29 (25.99 %)

2.6 Related Work

2.6.1 Models of Interaction

There are two general models of information access: a traditional IR model oriented towards query refining and interaction models describing user interfaces that focus on how the user searches for information.

Hearst [49] describes a model of the first class as a constant process of refining the original query. The user starts with an information need, she forms a query, sends it to the system, receives results, evaluates them and reformulates the query to improve it. The process continues until the perfect answer is found. Most of IR systems operate using this model. It describes the main components of the interaction and focuses the designer on two aspects: search and query reformulation. It seems to hold well in real world settings. The model is primarily focused on getting the answer from the system and does not explain how and why the user interacts with it.

That model assumes that the information need remains static, that users need all and only the documents relevant to that need. It assumes that users can correctly recognize the relevant documents and the only activity is to type in a query, read the list of results and mark the relevant documents.

The second class of models focuses on *how* the users interact with the system. They attempt to take into account that users learn while interacting. Bates [12] proposed the "berry-picking" model of information retrieval with several key points. First, the users information need, and consequently their queries, is shifting as they interact with the system. A new datum can make the user go off in an unexpected direction, forcing him to adjust his expectations about the document and database contents. Second, the model states that the user's information need is satisfied not by a single set of documents but by bits and pieces of information the user acquires during his interaction process.

The study by O'Day and Jeffries [76] supported the "berry-picking" model. They found that information seeking is not one single search process, but a set of diverse but interconnected searches on the problem. The main value of that search process is the information and learning experience accumulated during the search instead of the final set of documents.

Our model of interaction closely resembles the former approach as we define a specific task that both the user and the system are trying to achieve. However we also emphasize a strong role of the user in the interaction process. We attempt to determine the most efficient way for the user to interact with the information organization system as a part of our analysis. That is where our research touches the area of user modeling.

2.6.2 User Modeling

The application idea behind user modeling is as follows: if the system can create an accurate representation of the user, it can anticipate and predict the user's actions and needs with greater reliability and therefore make the interaction process more effective. User modeling is considered to be an inherent part of information retrieval [15]. Generally there are two parts to the modeling: a system and a user. Consequently all study in User Modeling in IR could be viewed as either system or user oriented.

System-centered techniques include relevance feedback, where the user is modeled by texts that she judges as relevant (or not). This data is used to retrieve or discard similar texts [93]. The second approach is query expansion. The initial or modified query is used for user modeling. Terms and logical connectives are expanded, contracted, added, switched, etc., by automatic, semi-automatic, and manual techniques [36]. Another method is to build into the system ways and means by which user can model their problems on their own with some assistance from the system [30]. Harman [44] studied different query expansion techniques that adopt terms generated by relevance feedback. She simulated the user selecting the terms for expansion by filtering in only the terms that appear in relevant documents unseen by the user.

The third method, Selective Dissemination of Information (SDI), was developed by Hans Peter Luhn [70]. There are several other names for this method, one of which is routing. The user model

is represented by a search statement or user profile that is used for filtering the text from oncoming streams like newswires. Profiles are dynamic and change with time.

There are far fewer studies on user-centered techniques. The user modeling is generally accomplished through various interview and analysis techniques [47]. An alternative is to build a cognitive model and analyze how well it corresponds to the real world experience [8].

Many different techniques are used for User Modeling. They range from Bayesian methods [28], neural networks (e.g., to summarize user's WWW navigation behavior [43]), stereotypes (e.g., for ascription of WWW-related interests [10]) and logic-based techniques [40].

2.6.3 Evaluation of Interactive Systems

There are several different ways of evaluating interactive systems. A user study is probably the most widely employed method. Here a person is involved in applying the system to a number of tasks. The user's performance is measured and used as the quality estimate. An example of a user study is the interactive track included in TREC [46]. It is a good example of how the "overall" performance of both the user and the system working together is measured. User studies are also applied to evaluate some particular aspects of the system or used indirectly for system analysis. For example, in their post-experimental analysis of the TREC interactive experiments Hearst and Pedersen [50] discovered that their Scatter/Gather clustering system generates good cluster descriptions – 75% of the users were able to choose the cluster with the largest number of relevant documents using the textual summaries the system created.

User studies are usually very expensive, time-consuming and difficult to execute. Designing a good and informative user study is almost a work of art. With all the great information organization and visualization systems developed and evaluated in recent years few user studies have produced any conclusive evidence showing the difference between a newly designed system and more traditional approaches [58, 89, 56]. Those that do require imposing significant constraints to limit the amount of uncertainty in the experiment. Our evaluation approach includes a user study. In contrast to past studies, we apply it to individual aspects of the system instead of attempting to measure the overall performance. This specialization simplifies the user study, focuses it, eliminates significant amount of uncertainty, and boosts the statistical quality of the results.

A different evaluation approach is the predictive evaluation method (e.g., see Card and Morgan [23]). This technique estimates how fast a particular task can be executed using the system. It requires the task to be defined precisely and the system is evaluated particularly for this task. This is achieved by subdividing the task into a number of unit actions such as key presses and mouse clicks, where the time necessary to perform the unit actions is known. A set of possible strategies that combine the unit actions together is generally assumed for the user. This is similar to our approach as we are interested in how fast the user can locate all relevant information and we also assume different strategies for the user. However, we are interested in measuring the amount of data the user is forced to analyze before finding all relevant documents and not the actual time that is required to complete the search.

2.6.4 Performance Measure

There are different evaluation statistics for measuring how well the system is able to support a retrieval task. Generally, the designers of information retrieval systems strive to maximize the expected proportion of relevant documents returned to the user – *recall* – and the proportion of the relevant documents among the retrieved material – *precision* [45, 46]. These statistics have both advantages and disadvantages. Many alternatives were suggested.

For example, Dunlop [35] introduces the notion of number-to-view graphs, where the number of relevant documents a user needs from the system is plotted against the number of documents the user would have to view to encounter them. He also considers time-to-view graphs where the same number of relevant documents required by the user is plotted against the time the user has to spend with the system to retrieve them. This allows both engine and interface performance be discussed in the same evaluation framework.

Tague and Schultz [97] define an evaluation model that consists of answers for two following questions: What are the descriptions and measures that will characterize the system? How can the contribution of individual system components be isolated? The authors measure system performance in terms of informativeness, user friendliness, and contact time with the system. They exemplify the model by applying it to evaluation of an online catalog interface.

Evaluation statistics well suited for an off-line retrieval task do not measure well in interactive settings. Su [94] considers several criteria and measures for evaluating interactive IR performance. Six professional searchers served as intermediaries for 40 subjects. They performed searches in an online search system in a large university. 20 different measures were studied. The author finds that the value of the search results as a whole is the best single measure for IR performance. Effectiveness and interaction account for 72% of the variance, and the interaction is more important than effectiveness. She also states that precision is not a good indicator of IR performance. She defines interaction in terms of user's confidence in completeness of search results, success of interview in question and searcher's knowledge about use of databases for the question.

2.6.5 Ranked List Visualization

The documents retrieved by the system are generally presented as a list of titles ordered by increasing rank. The title is often accompanied by a subset of the document metadata, such as date, source, and size of the text. A numeric score assigned to the each document by the system together with the actual rank value is also attached. This information is called a *document surrogate* [106].

Some implementations of the ranked list presentation provide a short summary or a description of the document that are either human-built (e.g., as in the bibliographic catalogues [2]) or automatically generated [54].

A number of systems visualize the information about query term occurrences in the documents. For example, the Google search engine [42] presents a *keyword-in-context* document surrogate. It displays several sentences or short fragments of the document text that contain the query terms together with the document title in the ranked list.

The TileBars [48] system shows a rectangular bar attached to each document title. The bar is subdivided into columns corresponding to passages in the document (the length of the bar is proportional to the number of passages). The bar is also divided into rows corresponding to individual query terms. Each cell – the intersection of a row and a column – is shaded with a dark color with intensity proportional to the term frequency of that query term in the particular passage. The darker the shade the greater the value. Documents with query terms overlapping in the same passage are more likely to be relevant than those where the terms are scattered through the document.

The system designed by Byrd and Podorozhny [21] computes the influence of each query term in retrieving a particular document or the weight for that term in the document. It displays the weights as colored bars side by side attached to the document title in the ranked list. Each color corresponds to a query term and the length of the bar is proportional to the term weight value in the document. That visualization allows the user to isolate the documents with all query terms (i.e., that are hopefully more relevant) from the documents that contain only a query subset. For example, given a two word query the visualization will immediately highlight the documents where the two words occur together from the documents that contain only the first word, albeit with a very high frequency.

2.6.6 Relevance Feedback Interface

A standard interface for the relevance feedback is to provide a checkbox or a choice control by the document title that allows the user to indicate the relevance value of the document [14]. However, collecting multiple relevance judgments is very effort consuming process. Several web-search engines adopted a one-click approach by attaching a "more like this" link to the document surrogate [42]. Users following that link are in fact performing a relevance feedback algorithm with that one document marked as relevant.

The relevance feedback term selection and weighting mechanism can be very complicated and difficult to explain. For most of the time the algorithm works in a "black box" – the user submits her

judgments about the documents and the algorithm produces a new query without much explanation. Koenemann and Belkin [56] studied several variations on the relevance feedback algorithm that allowed users different levels of control on the term selection process. The first system (*opaque*) did not allow any control for term selection, the second system (*transparent*) allowed the user to see the terms that were added, and the third (*penetrable*) allowed users to decide which terms suggested by the system shall be added to the query. The users were much more successful with relevance feedback than without it. The *penetrable* system required significantly less time to achieve better queries and the users generally preferred it over the other designs.

2.7 Summary

In this thesis we conduct an exploratory analysis of two information organization systems. What distinguishes our study from previous work in information visualization and information retrieval is that we consider the information organization an integral part of the information retrieval process. We analyze the effectiveness of the information organization system for the most specific retrieval task – helping the user to locate the relevant documents.

The second distinctive quality of our work is the manner in which we conduct the analysis. We formalize the interaction process between a user and a system and focus our attention on the information that the system presents to the user. We introduce the Strategy Based Evaluation framework that allows us to study if the system *can* support an arbitrary user in solving the retrieval task. The analysis is based in part on simulating a user with an algorithmic search strategy and conducting a number of experiments on a standard dataset.

An evaluation of an interactive system often involves real people working with the interactive system, i.e., a user study is conducted [96]. People are very different in their abilities and skills. It is very difficult to control all the variables involved. An accurate user study may require a large number of participants to average out individual qualities of each user. However, experiments even with a small group of people are very expensive and time-consuming. In contrast, our approach provides absolute control for the researcher and it is relatively cheap as most of the analysis is done off-line. The limited user study that is required to validate the information clues is done much faster and in a highly more controlled environment than a full-blown study of the complete system.

Such an analysis is inexpensive, efficient, and gives a deeper understanding of how the system works. The third quality of our work is the insight into the actual properties of these information organization systems.

CHAPTER 3

DOCUMENT CLUSTERING

In this chapter we describe and analyze an information organization system that is based on a hierarchical agglomerative clustering algorithm. The system groups the retrieved material into a number of clusters that completely cover the document set. We conduct the analysis following the Strategy Based Evaluation framework. Recall from Section 2.3 that the SBE analysis consists of five parts including experimental task, system design, search strategy, evaluation measure, and experiments.

3.1 Experimental Task

The task we are trying to solve with the document organization system is locating relevant documents among the retrieved material as quickly as possible. In Section 2.3.1 we described two possible initial conditions for our experiments. The first is the no-information condition which does not provide the search strategy with an obvious starting point in its exploration. In that case no relevant judgments for the retrieved documents are made available to the search strategy. The second is the first-relevant initial condition which assumes that the highest ranked relevant document and all the non-relevant documents that precede it in the ranked list are marked with their corresponding relevance judgments. This provides the search strategy with a good seed of useful information.

The first experimental question we study in this chapter is how the available relevant information affects the performance of the search strategy. We compare it to the performance numbers for the ranked list and the relevance feedback search strategies. Practically, this is the question of whether we can locate the first relevant document in the set more quickly by using the clustering document organization than by following the ranked list.

3.2 System Design

The system in this chapter brings similar documents together into groups and presents these groups as described in our example in Section 1.1. There are two important parts of the design: arranging the documents into clusters and presenting the clusters on the screen.

3.2.1 Clustering Algorithm

A hierarchical agglomerative clustering algorithm creates a hierarchy of clusters – it builds a tree where each node is a cluster of objects and the clusters corresponding to the node's immediate children form a complete partition of that cluster [73]. On input the algorithm receives a set of objects and a matrix of inter-object distances. It starts by assigning each object to its own unique cluster – the leaves of the future tree. The algorithm iterates through the cluster set by selecting the closest pair of clusters and merging them together forming a new cluster that replaces them in the cluster set. A node corresponding to this new cluster is created in the tree and the selected pair of clusters become its children. That procedure is executed until all objects are contained within a single cluster, which becomes the root of the tree.

This is a general algorithm that is instantiated by choosing a specific distance function for clusters. Indeed, the distance between a pair of singleton clusters is well-defined by the original distance matrix. If one of the clusters contains more than one object, the inter-cluster distance is determined by a specific heuristic. For example, we may define the inter-cluster distance as the

Table 3.1. Lance-Williams coefficients for most known agglomerative clustering methods. n_i denotes the size of the ith cluster.

Method	α_i	α_i	β	γ
Single linkage	0.5	0.5	0	-0.5
Complete linkage	0.5	0.5	0	0.5
Group average	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
Weighted group average	0.5	0.5	0	0
Centroid	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	$\frac{-n_i \cdot n_j}{(n_i+n_j)^2}$	0
Ward	$\frac{n_i+n_k}{(n_i+n_j+n_k)}$	$\frac{n_j+n_k}{(n_i+n_j+n_k)}$	$\frac{-n_k}{(n_i+n_j+n_k)}$	0

smallest distance between two objects in both clusters. Other suggested alternatives include the average distance between two objects and the maximum distance.

Creating a new cluster by merging requires the distance matrix to be recalculated to establish the distances between the new cluster and the rest of set. That calculation could be very time consuming if the calculations were to be done using the initial distance matrix. Lance and Williams [59] have shown that many different clustering methods can be derived from the following equation and computed quite efficiently:

$$d_{k,i\cup j} = \alpha_i \cdot d_{k,i} + \alpha_j \cdot d_{k,j} + \beta \cdot d_{i,j} + \gamma \cdot |d_{k,i} - d_{k,j}| \tag{3.1}$$

here, $d_{k,i\cup j}$, the distance between the cluster created by merging ith and jth clusters and an arbitrary cluster k is defined as a nonlinear function of distances between the individual clusters. The coefficients for the most commonly used methods are presented in Table 3.1.

In this thesis we consider six different clustering techniques based on the generalized agglomerative algorithm (Table 3.1). The *single linkage* method defines the distance between two clusters as the smallest distance between two objects in both clusters. The *complete linkage* uses the largest distance instead. These two methods represent two extremes of the generally accepted requirement that the "natural" clusters must be cohesive and isolated from the other clusters [73]. Single linkage clusters are isolated but not cohesive, while complete linkage produces cohesive groups that may not be isolated at all. The other four methods represent some compromise between the two extremes.

In the *centroid* method a cluster is represented by a centroid, i.e., the the center of gravity. The distances between the clusters are the distances between the corresponding centroids.

For both *group average* and *weighted group average* the distance to a newly created cluster $d_{k,i\cup j}$ is a weighted sum of the distances to the existing clusters $d_{k,i}$ and $d_{k,j}$. For the group average method these individual distances are normalized by the cluster sizes, so a large cluster will have more influence on the resulting distance. For example, if $n_i > n_j$, $d_{k,i\cup j}$ will resemble $d_{k,i}$ more closely than $d_{k,j}$. In the case of the weighted method both $d_{k,i}$ and $d_{k,j}$ have the same weight. Thus both merging cluster have the same influence on the result. The idea is to enhance the weight of a small cluster as it joins a large group.

The *Ward* method [102] is based on minimization of clustering "expansion." Attempting to merge two clusters the algorithm computes the central point of the future cluster and then calculates the total sum of squares of distances from that point to all objects in that future cluster. A cluster pair that produces the smallest sum is then selected for actual merging. Note that contrary to all other approaches this distance between two clusters cannot be interpreted in terms of the initial inter-object distances.

Thus the second experimental question we examine in this chapter is the comparison of six different clustering algorithms to determine which is best suited for the task of helping the user to quickly locate relevant documents.

3.2.2 Creating Partition

The hierarchical agglomerative clustering algorithm produces a hierarchy of clusters. In this thesis we study an approach which presents the user with a partition of the document set – a set of clusters that divides the retrieved material into the groups of similar documents. An effective visualization of a hierarchy is a more difficult and complex task than showing a flat partition. We believe that understanding and navigating a cluster hierarchy is a more challenging and unnecessary interface task for the user. There are more chances to confuse and distract him from the task of locating relevant information.

To create a partition of the document set from a cluster hierarchy we "cut" the hierarchy at some level, i.e., stop the clustering algorithm before it reaches the root of the tree. The clusters present in the set at that moment form the required partition. The problem is to decide at what point to make the cut. For example, the Scatter/Gather research [50] fixes the number of clusters in the document set. We, on the other hand, set a threshold on the similarity distance between clusters – while iterating through the cluster set the algorithm stops as soon as the distance between the closest pair of clusters exceeds the threshold. If the threshold is kept constant from session to session, the density of the clusters becomes the system's invariant. The user will always know what minimal degree of similarity to expect from the documents placed in the same cluster.

To select the threshold value we conduct our experiments following a basic two-way cross-validation scheme. We divide the experimental data set into three parts: training, testing, and evaluation data. Recall from section 2.4 that our data consists of documents from two different collections (TREC-5 and TREC-6) retrieved by queries of four different types – eight different data sets. Each data set serves as a separate training data set – we exhaustively search for the threshold value that produces the best average performance on that data set. The training phase produces four potentially different threshold values for the documents from one collection, i.e., one threshold for each query type. We select one threshold value out of these four that gives us the best performance on all data sets from the same collection: the training data set and the other three data sets of documents retrieved from the same collection combined is the testing set. The selected threshold value is used to organize the documents from the other collection. Thus the other four data sets form our evaluation group. In the section detailing our experimental results below we report only the numbers from the corresponding evaluation data sets.

3.2.3 Presenting Clusters

A clustering algorithm brings together similar documents. We show the document set to the user as a list of clusters where each cluster in turn is arranged as a list of document titles. We call this representation the *clustered list* by analogy with the ranked list. This presentation is similar to the example given on Figure 1.3 and several past studies adopted similar approaches [50, 64]. However, we do not show any textual descriptions for the clusters. Instead we select one representative document from each cluster and place it at the top of the cluster's list. This document is supposed to be the most helpful to the user in establishing the overall relevance value for the cluster. Ideally, by looking at the representative document the user should be able to decide whether to examine the cluster or skip it and go to the next one. The rest of the documents inside the cluster are kept in their original order – they are ordered by the probability of being relevant to the user's request. This should insure an effective ranking [100, p.88]. The clusters are ordered using the original rank of the highest ranked document in each cluster.

The third experimental question we study in this chapter is the choice of the representative document or the first document in a cluster list. We consider four different alternatives. The first, a rather obvious choice is to use the document that is the best representation of the cluster – the cluster centroid, or the document that is the most similar to the actual centroid. The second alternative we consider is the highest ranked document in the cluster. Our intuition is that if this document is non-relevant than the rest of the cluster is very likely non-relevant. The third choice is to use the lowest ranked document: if that document is relevant than it is very likely that the rest of the cluster is also relevant.

The documents at the top of the ranked list most likely are relevant and the documents at the bottom of the list most likely are non-relevant. Lewis [66] speculated that the best way to find the boundary between the relevant and non-relevant material in the list is to examine the documents in the middle. The last candidate for the cluster representative is the medium ranked document – the document whose original rank is the median of the cluster.

There are two different information clues that the system supplies to a user. The first, two documents are similar to each other if and only if they are listed in the same cluster. The second, like in the original ranked list, both the documents and clusters are ordered by their importance. The user should start at the top of the list and follow it down. The only difference from the ranked list is that the user can abandon examining a cluster without looking at every document in it and jump to the next cluster in the list. In the next section we consider when to take this action.

3.3 Search Strategy

The use of clustering in Information Retrieval is tightly linked with the Cluster Hypothesis of Information Retrieval: "closely associated documents tend to be relevant to the same requests" [100, p.45]. A simple corollary from this hypothesis is that documents similar to a known relevant document are likely to be relevant as well.

The search strategy we define for this document organization system targets the documents that are most likely to be relevant. It selects the document with highest similarity to the known relevant documents and at the same time it minimizes the similarity to the known non-relevant documents:

$$F_{cl}(\mathcal{D}_t, d) = \theta_1 \cdot \sum_{x \in \mathcal{R}_t} sim(x, d) + \theta_2 \cdot \sum_{x \in \mathcal{N}_t} sim(x, d)$$

where $sim(x, d)$ is the similarity between two documents, $\mathcal{R}_t$ and $\mathcal{N}_t$ are the sets of all examined relevant and non-relevant documents at time step t.

In a clustering partition of a document set the inter-document similarity is represented as a binary relationship: the documents are either similar to each other – they are placed into the same cluster – or they are deemed different by the algorithm and placed in separate clusters:

$$sim(x, d) = \begin{cases} 1, & \text{if } x \text{ and } d \text{ are in the same cluster} \\ 0, & \text{otherwise} \end{cases}$$

At each time step the search strategy selects the document d with the highest value of $F_{cl}(\mathcal{D}_t, d)$. If two documents from different clusters have the same score, the search strategy prefers the document from the higher ranked cluster. For two documents from the same cluster the ties are broken by selecting the document with the highest place in the cluster's list.

The fourth question that we consider in this chapter is the importance of the relevant and non-relevant information while computing the search strategy function. We experiment with different values for θ_1 and θ_2 and compare the performance of the resulting search strategies.

3.4 Evaluation Measure

Section 2.3.4 introduces the average precision as our evaluation measure in this work. In the section that follows we present average precision numbers computed for the whole retrieved document set.

3.5 Experiments

3.5.1 Initial Conditions

The first question is to compare the performance of the search strategy F_{cl} with the performance of the ranked list and relevance feedback search strategies. The last four columns of Table 3.2 show the average precision values for F_{cl} and the percentage difference from the ranked list ("RL"). The

Table 3.2. Performance of F_{cl} search strategy on document set partitions created by the group average algorithm. Average precision numbers for two initial conditions and percent improvement over the original ranked list ("RL") are shown. The table also includes the precision values for the ranked list and interactive relevance feedback search strategies. The latter approach used 100 terms form query expansion.

Data set		RL	RF	group average	
		Original	+100t	no-info (v. RL%)	first-rel (v. RL%)
TREC-5	Full	38.49	41.87	45.48 (18.16 %)	44.14 (14.67 %)
	Desc	36.06	44.19	41.40 (14.82 %)	40.84 (13.26 %)
	Title	33.19	36.29	36.81 (10.91 %)	37.38 (12.64 %)
	ExpTtl	29.91	32.64	36.63 (22.45 %)	33.43 (11.75 %)
TREC-6	Full	53.67	54.48	57.69 (7.47 %)	56.73 (5.69 %)
	Desc	46.66	58.54	54.69 (17.21 %)	53.86 (15.44 %)
	Title	46.19	56.23	51.78 (12.12 %)	52.56 (13.80 %)
	ExpTtl	51.64	53.12	54.61 (5.76 %)	54.34 (5.23 %)
total average		41.98	47.17	47.39 (13.61 %)	46.66 (11.56 %)

search strategy worked with clustering structures built by the group average algorithm. The highest ranked document was used as the cluster representative. The search strategy function weights were set to $\theta_1 = 1$ and $\theta_2 = -1$.

We observe a small difference in average precision due to available relevant information. The last two columns ("first rel") show the precision for the search strategy F_{cl} beginning with the highest ranked relevant document marked in the retrieved set. The preceding two columns ("no rel info") show the same performance but without any relevant data available *a priori*. The former values are slightly smaller than the later ones on average. It shows that one can locate the first relevant document in the set slightly faster with the clustering organization than by following the ranked list. However the difference is small and not statistically significant.

The average precision numbers for the ranked list and interactive relevance feedback are copied from Table 2.3. We observe 13.61% and 11.56% improvement over the ranked list. These differences are statistically significant. The differences between F_{cl} and the search strategy for interactive relevance feedback ("RF +100t") are small and not statistical significant.

3.5.2 Algorithms

The second experimental question is the comparison of six different clustering algorithms. Table 3.3 shows the corresponding average precision values for single linkage, complete linkage, group average, weighted group average ("weighted average"), centroid, and Ward algorithms. The highest ranked document was chosen as the cluster representative for each algorithm. The search strategy began with no relevant information. The weights for F_{cl} were set to $\theta_1 = 1$ and $\theta_2 = -1$.

We observed that the group average and Ward algorithms result in better performance consistently across all experimental variables considered in this chapter. The difference between these two algorithms is insignificant. Also the differences between the group average algorithm and both weighted average and complete linkage are not statistically significant. The single linkage method is a clear "loser" in this competition.

3.5.3 Cluster Representatives

The third experimental question is to evaluate the effect of the cluster representative on the quality of the document organization. Recall from Section 3.2.3 that the representative document is the one placed at the top of each cluster's list. It is supposed to be the most helpful in establishing the overall relevance value for the cluster. Ideally, by looking at the representative document the user should be able to decide whether to examine the cluster or skip it and go to the next one.

Table 3.3. Performance of F_{cl} search strategy on document set partitions created by six different clustering algorithms.

Data set		single linkage	complete linkage	group average	weighted average	centroid	Ward
TREC-5	Full	43.00	45.42	45.48	44.51	43.42	45.28
	Desc	39.59	41.15	41.40	40.70	40.05	41.91
	Title	36.12	36.60	36.81	36.80	36.24	37.14
	ExpTtl	35.15	36.59	36.63	35.82	34.77	36.97
TREC-6	Full	56.16	57.30	57.69	56.06	56.35	56.25
	Desc	52.27	53.24	54.69	54.08	52.91	54.20
	Title	50.40	52.89	51.78	51.78	52.21	52.47
	ExpTtl	54.52	54.70	54.61	54.11	54.77	54.06
total average		45.90	47.24	47.39	46.73	46.34	47.28

Table 3.4. Performance of F_{cl} search strategy on document set partitions created by the group average algorithm using four different types of cluster representative documents.

Data set		centroid	highest ranked	lowest ranked	medium ranked
TREC-5	Full	44.51	45.48	38.92	43.62
	Desc	39.72	41.40	38.51	39.81
	Title	36.64	36.81	36.21	37.11
	ExpTtl	35.73	36.63	35.63	35.36
TREC-6	Full	54.63	57.69	52.89	53.63
	Desc	52.35	54.69	50.71	52.20
	Title	51.45	51.78	47.98	47.79
	ExpTtl	51.08	54.61	49.39	48.07
total average		45.76	47.39	43.78	44.70

Table 3.4 shows the average precision values obtained for four different cluster representatives. We used the document organization structure created by the group average algorithm. The no-information initial condition – was used to initialize the search strategy. The relevant and non-relevant weights for F_{cl} were set to $\theta_1 = 1$ and $\theta_2 = -1$.

The choice of the highest ranked document in a cluster as the cluster's representative is the most effective for the task of locating the relevant material. We observe an almost 4% drop in average precision while selecting the document closest to the cluster's center. The difference is statistically significant. Our explanation is that the highest relevant document allows user to quickly discard non-relevant clusters – if even the highest ranked document in the cluster is non-relevant, then it is very likely that the rest of the cluster is also non-relevant.

3.5.4 Search Strategy

The fourth experimental question we consider in this chapter is the optimization of the search strategy function F_{cl}. Table 3.5 shows the performance of the search strategy on the clustering structures created by the group average algorithm using the highest ranked document as the cluster representative. The search strategy has no relevant information available to it at the beginning of the search.

The six columns in Table 3.5 correspond to six different value settings for the relevant and non-relevant weights (θ_1 and θ_2) in F_{cl}. The column titles are presented in the form θ_1:θ_2. There is a clear maximum in performance of the total average for $\theta_1 = 1$ and $\theta_2 = -1$. Increasing or decreasing the

Table 3.5. Performance of F_{cl} search strategy on document set partitions created by the group average algorithm. Average precision values are shown for different ratios of relevant and non-relevant weights (θ_1 and θ_2) in F_{cl}.

Data set		Relevant to non-relevant weight ratio					
		1:0	1:-0.25	1:-0.5	1:-1	1:-2	1:-4
TREC-5	Full	38.24	45.14	45.40	45.48	44.71	44.35
	Desc	38.19	40.55	40.99	41.40	40.74	40.57
	Title	34.55	36.70	37.02	36.81	36.67	36.64
	ExpTtl	30.82	35.83	36.30	36.63	36.59	36.20
TREC-6	Full	54.31	56.99	57.03	57.69	57.51	57.76
	Desc	50.93	54.36	54.45	54.69	53.12	53.46
	Title	49.49	51.94	51.98	51.78	52.90	52.47
	ExpTtl	49.25	54.27	54.31	54.61	54.86	55.26
total average		43.22	46.97	47.18	47.39	47.14	47.09

value for the non-relevant weight θ_2 from -1 leads to lower values of average precision. All pairwise differences between the maximum ("1:-1") and the other parameter sets are statistically significant.

3.5.5 Knowing the First Relevant Document

The last experiment we describe in this chapter is locating the relevant information given that some of the relevance data is available to the system. In Section 3.5.1 we have considered the performance of the search strategy F_{cl} for both initial conditions including the case when the highest relevant document is marked at the beginning of the search. In that experiment we selected the threshold values to maximize precision over all retrieved documents. In this section we maximize and record precision only over the unexamined part of the document set and excluding the initial information.

The last four columns Table 3.6 show the performance of the search strategy F_{cl} on document organizations created by the group average and Ward algorithms and the percentage improvement over the search strategy for the original ranked list. The highest ranked document was selected for each cluster as its representative and the relevant and non-relevant weights in F_{cl} were set to $\theta_1 = 1$ and $\theta_2 = -1$.

We see that the clustering visualization is a much more effective method for locating relevant documents than the ranked list if we already found one relevant document. This confirms the results from Section 3.5.1. There is an increase in average precision of 20.3% versus 11.6% we observed in the previous experiment. The ranked list does a very good job at placing the first relevant document close to the top, while the rest of the relevant material is scattered farther down the list. That is why when we include the documents from the first-relevant document subset into calculating the average precision, the difference is smaller. The clustering organization brings the relevant documents together, so locating one of them speeds up finding the rest. It is equivalent to the relevant material ending up pulled closer to the beginning of the ordering created by the search strategy.

In contrast to the results in Section 3.5.1 the interactive relevance feedback does a better job at locating relevant documents than our search strategy once one relevant document is examined. The differences are -5.2% and -3.9% for the group average and Ward algorithms. These differences are statistically significant. However, the relevance feedback is much more computationally expensive. In fact it took us at least 1000 times longer to run the experiments using the interactive relevance feedback search strategy than the clustering search strategy. If we compare the performance of the clustering search strategy to the interactive relevance feedback that uses just 10 terms to expand the query (see Table 2.4), we see that the numbers are similar and the differences are not statistically significant.

Table 3.6. Performance of F_{cl} search strategy on document set partitions created by the group average and Ward algorithms. A minimal amount of the relevant information is available at the beginning of the search. Average precision numbers and percent improvement over the original ranked list ("RL") are computed excluding the documents from the first-relevant document subset. The table also includes the precision values for the ranked list and interactive relevance feedback search strategies. The latter approach used 100 terms for query expansion.

Data set		RL	RF +100t	group average (v. RL%)	Ward (v. RL%)
TREC-5	Full	36.17	44.17	45.03 (24.51 %)	46.02 (27.22 %)
	Desc	34.09	50.60	42.07 (23.41 %)	44.73 (31.23 %)
	Title	30.95	41.39	37.68 (21.75 %)	36.92 (19.29 %)
	ExpTtl	30.95	38.26	38.55 (24.55 %)	40.95 (32.29 %)
TREC-6	Full	51.02	54.70	56.67 (11.07 %)	56.74 (11.20 %)
	Desc	42.48	60.39	52.53 (23.66 %)	53.01 (24.79 %)
	Title	40.10	54.74	49.73 (24.01 %)	49.97 (24.61 %)
	ExpTtl	47.21	50.07	51.54 (9.17 %)	50.94 (7.89 %)
total average		39.12	49.29	46.73 (20.27 %)	47.41 (22.31 %)

3.6 Discussion

We see that clustering can greatly improve the effectiveness of the ranked list. In fact it can be almost as effective as the interactive relevance feedback based on query expansion. Surprisingly this high performance can be achieved by following a very simple strategy. Given a list of clusters created by the group average algorithm, a user starts at the top of the list and follows it down examining the documents in each cluster. As soon as she sees that a cluster has more non-relevant documents than relevant ones, she discards that cluster and switches over to the next one. Our experiments show that her performance will be as good as that described in Section 3.5.1.

Another nice quality of the clustering approach is the well-defined informational clues it uses. The preferred order in which documents should be examined is outlined by the clustered list. The inter-document similarity is also easily determined from the list. We believe these informational clues to be clear, obvious, and based on the skills possessed by anyone who has used a web search engine. We do not test the understandability of the clues in our experiments.

We have observed that the threshold value can significantly affect the effectiveness of the system. In these experiments one threshold was used for all document sets. We believe the performance may be improved if the threshold value is adapted to individual user requests. We considered using the threshold based on the query complexity. Instead of selecting the best threshold on one collection overall we used the best threshold for the individual query types, i.e., after we find the best threshold for the TREC-5 Title queries we apply it to cluster the documents from the TREC-6 Title queries only. The average precision numbers were slightly worse than those that we have presented in this chapter.

The clustering algorithm creates a non-overlapping partition of the document set with sharp boundaries: each document is assigned to only one cluster. A document discussing several topics can be clustered based on any one of them. This way a group of non-relevant documents discussing one topic can attract a relevant document if it happens to contain that non-relevant topic as well as the relevant subject. In fact, our observations suggest that the clusters created by the algorithms contain a mix of relevant and non-relevant documents – pure relevant clusters (clusters with no non-relevant material) are generally rare.

The relationship between documents is a uniform one: the documents are either in the same cluster or they are in different clusters. This representation carries no clues about how documents from different clusters relate to each other. For example, if we find a relevant document in a generally non-relevant cluster, we might be interested to look at a document that is similar to it

but placed outside of that cluster. In the next chapter we consider a document organization system that eliminates all these limitations by sacrificing the simplicity of the representation.

3.7 Related Work

Both document and word clustering have been studied in the field of information retrieval for several decades. Willett [104] gives an excellent overview of the existing algorithms and applications. The use of clustering is based mostly on the Cluster Hypothesis: "closely associated documents tend to be relevant to the same requests" [100, p.45]. The document clustering has been intensely studied in the context of improving the search and browsing performance by pre-clustering the entire collection [104, 32, 31].

Croft [29] and more recently Hearst and Pedersen [50], showed that the Cluster Hypothesis also holds in a retrieved set of documents. However, they did not study how the clustering structure may help a user to find relevant information more quickly. In contrast to those studies Voorhees [101] could not find any conclusive support for the Cluster Hypothesis.

Numerous studies and anecdotal evidence hint that document clustering can be a better way of organizing the retrieval results. We could not find any strong experimental results that support this assumption. The Northern Light [75] and Dataware search systems [33] are two examples of on-line commercial systems that organize the search results into folders. Generally such systems operate with full-text documents, MetaCrawler-STC [109] at the University of Washington places the web search results into overlapping clusters based on the snippets returned by the engine.

There exist many different clustering algorithms and a particular choice is usually motivated by the system design. Most often the algorithm efficiency is the crucial point. If we are to cluster a collection of hundreds of thousands of documents then using an algorithm that requires us to make $N \cdot (N-1)/2$ pair-wise comparison for N documents would be prohibitively expensive. A number of alternative solutions have been developed. For example, Scatter/Gather [31] interacts with a user – divides or merges clusters per user's request – in a constant time due to a clever near linear ($O(kn \log n)$) preprocessing phase.

Another popular approach is K-means clustering. The number K is a parameter of the algorithm and it determines the number of final clusters. The algorithm starts by defining K cluster centroid or "seeds" and then compares every objects with every centroid. The object is assigned to the cluster with the closest seed. The execution time is $O(K \cdot N)$.

These efficient algorithms sacrifice some accuracy in arranging the documents to receive a significant advantage in speed. This is achieved by ignoring some of the inter-object similarity information. If the number of documents is small, it is more cost-effective to employ $O(N^2)$ techniques that make use of all inter-object similarity information – the whole $N \cdot (N-1)/2$ set of pair-wise distances. This is the reason why we consider a system built around a hierarchical agglomerative clustering algorithm.

One important problem with all clustering approaches is that the parameters that define the final set of clusters have to be determined *a priori*. For example, for K-means clustering one has to define the final number of clusters K. The agglomerative clustering builds a partition by joining together the most similar clusters requires a termination condition that determines where to draw the border between individual clusters.

The Scatter/Gather interface [50] presents the document clusters as text. It groups the documents into five (or any fixed preselected number) clusters and displays them simultaneously as lists. On a large enough screen, the top several documents from each cluster are clearly visible. Another text-based visualization is presented by Leuski and Croft [64]. Their method is similar to the one used by Scatter/Gather, but the number of clusters is based on a similarity threshold. Their display looks more like a standard ranked list because they can have an arbitrarily large number of clusters (limited only by the size of the retrieved set).

3.8 Summary

While the algorithms considered in this study are well-known, we are the first to conduct such a thorough analysis of the clustering approaches in the context of organizing the documents retrieved by a search engine. We are the first to experimentally show that the clustering can be almost as effective as the traditional interactive relevance feedback in helping a user to locate the relevant material among the retrieved documents. At the same time clustering retains an important advantage over the relevance feedback methods based on query expansion – it provides the user with an important sense of control over the feedback process.

CHAPTER 4

SPRING-EMBEDDING VISUALIZATION

In this chapter we describe and analyze an information organization system that is based on the spring-embedding visualization algorithm. The system presents the retrieved documents as spheres placed in 2- or 3-dimensional space and positioned in proportion to the inter-document similarity (Figure 4.1). In contrast to the system described in the previous chapter this system does not make any distinction between organizing the documents and visualizing them. In fact, the 2- or 3-dimensional configuration of spheres is the document organization created by the system. It does not build any clusters – if a user sees some spheres arranged in groups, that is just an artifact of the inter-document similarity. Simply put, it just draws the documents, illustrating any structure that is already present in the data. Assigning any meaning to the structure is the user's task.

This chapter is organized in two parts. In the first part we conduct the Strategy-Based Evaluation of the system, describing the design, defining a search strategy, and asking a number of questions about the system and the spatial proximity clues that it provides. The spring-embedding algorithm maps inter-document similarity onto spatial proximity between document representations in 2 and 3 dimensions. The spatial proximity is the only informational clue that the system supplies to the user. It is a very unconventional approach compared to the traditional ranked list. That is why in the second part of the chapter we describe a small user study that tests the understandability of spatial proximity as a metaphor for the similarity between objects.

4.1 Strategy-Based Evaluation

As described in Chapter 2 the SBE analysis consists of five steps: experimental task, system design, search strategy, evaluation measure, and experiments.

4.1.1 Experimental Task

Section 2.3.1 describes two possible initial conditions for our experiments. The first initial condition (the no-information condition) defines that there is no relevance information available at the beginning of the search. Nevertheless, the original ranked list search strategy, interactive relevance feedback, and even the search strategy for the clustering document organization have a clear and intuitive starting point – the top of the list. It is the top of the ranked list for the former two strategies and the top of the clustered list for the latter. In contrast, the spring-embedding visualization of documents – a cloud of spheres (Figure 4.1) – does not have such a well-defined starting point. Thus, in this chapter we focus solely on the second initial condition. We assume that we know the judgments for the highest ranked relevant document returned by the retrieval system and for all the non-relevant documents that precede it in the ranked list. These documents are marked in the visualization. The task is to find the rest of the relevant material given this starting configuration.

4.1.2 System Design

Section 2.3.2 describes how we represent the retrieved documents as vectors of terms using INQUERY's term weighting scheme. The document vectors occupy a very high-dimensional space where the number of dimensions is equal to the number of unique terms in the retrieved documents. A set of techniques under the generic name of Multidimensional Scaling (MDS) has been developed to present high-dimensional objects in just a few dimensions [19]. An MDS algorithm accepts a

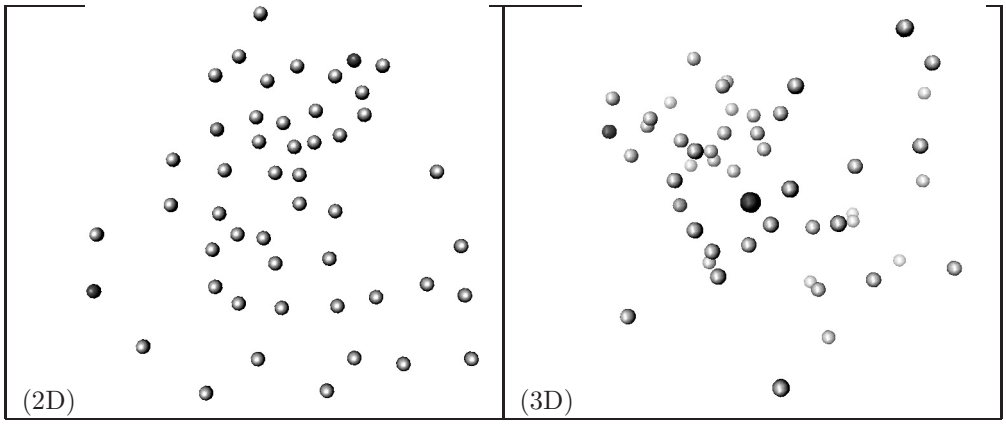

Figure 4.1. A set of 50 documents visualized in 2 and 3 dimensions.

matrix of inter-object dissimilarities and attempts to create a set of points in a Euclidean space such that the distances between the points as closely as possible correspond to the dissimilarities between original objects. A number of such algorithms exist; for our study we have selected the spring-embedding approach. Our choice was motivated by the graph-drawing heritage of spring-embedding [39, 96] – it is supposed to generate eye-pleasing pictures – and the availability of the source code.

The spring-embedding algorithm models each document vector as an object in 2- or 3-dimensional visualization space. It is assumed that the objects repel each other with a constant force. They are connected with springs and the strength of each spring is inversely proportional to the $1/cos$ dissimilarity between the corresponding document vectors. This "mechanical" model begins from a random arrangement of objects and due to existing tension forces in the springs, oscillates until it reaches a state with "minimum energy" – when the constraints imposed on the object placements by the springs are considered to be the most satisfied. The result of the algorithm is a set of points in space, where each point represents a document and the inter-point distances closely mimic the inter-document dissimilarity. Figure 4.1 gives an example of 50 documents "spring-embedded" in two and three dimensions. We call these spatial configurations *embeddings* of the document set.

Chalmers and Chitson [24] observed that if full-text document vectors are spring-embedded and all the pair-wise constraints are incorporated into the spring-embedding process (i.e., all springs are present in the model) the resulting structure is very tight and resembles a "soccer-ball". The intuition is that such a configuration is not desirable as it lacks any structure and therefore is not very informative. That research dealt mostly with spring-embedding of short article abstracts for which the "soccer-ball problem" does not apply. When attempting to extend the study to the full-text document they observed the problem but did not solve it.

In this study we adopt the approach suggested by Swan and Allan [96]. Facing the same problem, they introduced a threshold parameter into the algorithm: all constraints that fall below a predefined value are weakened significantly by squaring the corresponding document similarity. Since the similarity value is always between zero and one, that adjustment results in a much weaker spring connection. They decided to weaken the springs instead of simply removing them from the model since they observed drastic changes in the visualization as one spring was added or removed from the process. That threshold was initially selected randomly and the system users were allowed to adjust it during the interaction. They have not studied the problem of choosing the threshold automatically.

In section 2.3.2 we stated that we use the cosine of the angle between the vector ($\cos\theta$) to determine the similarity between documents. The spring-embedding algorithm requires a matrix of pairwise *dissimilarities* between documents as input. There are multiple ways one can define a dissimilarity function from the similarity function. For example, Swan and Allan used the sine

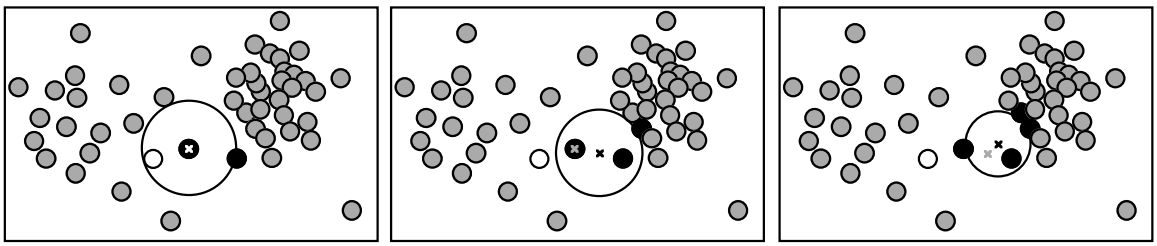

Figure 4.2. Three consecutive snapshots of our search strategy. We start from the document with an "X" inside and look for the rest of relevant documents. We show the state as the first, second, and third relevant documents are discovered. The white disks represent the known non-relevant documents, the black disks represent the known relevant, and the gray disks are the unknown documents.

of the angle ($\sin\theta$). Our past experiments showed that the quality of spring-embedded structures created using $\sin\theta$ is highly sensitive to the threshold value [62, 63]. In contrast using one over cosine ($1/\cos\theta$) creates better embeddings. Their quality is almost an invariant in relation to the threshold [61].

4.1.3 Search Strategy

We define a search strategy for the spring-embedding document organization system following the very successful approach outlined in the previous chapter (see Section 3.3). A corollary of the Cluster Hypothesis says that some of the relevant documents are very similar to the known relevant documents. The search strategy focuses on documents that are most likely to be relevant. At each time step it chooses the document with highest similarity to the known relevant documents:

$$F_{se}(\mathcal{D}_t, d) = -\frac{1}{|\mathcal{R}_t|} \cdot \sum_{x \in \mathcal{R}_t} dist(x, d)$$

where $F_{se}(\mathcal{D}_t, d)$ is the search strategy function for the spring-embedding apporach, $\mathcal{D}_t$ is a representation of the current state of the document set that includes the relevance judgments about the examined documents at time step t, d is an unexamined document, $dist(x, d)$ is the distance or dissimilarity between two document representations, $\mathcal{R}_t$ is the set of all examined relevant documents at time step t. At the beginning $\mathcal{R}_0$ consists of one document determined by the initial condition.

Figure 4.2 illustrates how the search strategy works. There the document spheres are shown as disks. The black disks indicate relevant documents, the white – non-relevant, and the gray – unexamined documents. The initially known relevant document is the disk with the cross in the center. The cross indicates the current origin or the center of the relevant document group. The big circle is centered at the origin and is the smallest such circle that can be drawn without intersecting a center of an unexamined document sphere. It highlights the sphere closest to the origin. We show three separate snapshots of the search strategy starting from the leftmost picture. The closest document to the origin is non-relevant (a white disk), it is ignored and the next disk is considered. It corresponds to a relevant document (a black disk), and, hence, is included into the cluster and the origin is adjusted. The cross is updated in the second snapshot and the big circle is redrawn; the gray cross indicates the old position of the origin. The process continues until all documents are examined.

In contrast to the F_{cl} strategy defined in the previous chapter, the F_{se} strategy does not use the information about the original ranking of the documents because it is not represented in the embeddings. It also ignores the examined non-relevant documents. In this chapter we wish to examine only the utility of using proximity to the relevant documents to isolate the relevant documents. In the next chapter we will look at more complex strategies.

The first experimental question we consider in this chapter is how well the F_{se} search strategy performs when compared to the search strategies for the ranked list and interactive relevance feedback.

4.1.3.1 Effect of Fewer Dimensions

The document vectors occupy a very high-dimensional space where the number of dimensions is equal to the vocabulary size of the retrieved set. When the documents are visualized with the spring-embedding algorithm some of the documents may be shown nearby when they are actually unrelated because of the constraints imposed by using fewer dimensions.

Additionally, it might not be possible to map the document dissimilarity onto Euclidean distance accurately if the dissimilarity function is not metric. For example, the triangle inequality is not always satisfied for the $1/cos$ dissimilarity measure:

$$\exists A, B, C \in D : \rho(A, B) + \rho(B, C) < \rho(A, C), \tag{4.1}$$

where $\rho(\cdot)$ is the dissimilarity function and D is the document set.

Thus, during the transition from the document vector space to a Euclidean space of a few dimensions the cluster of relevant documents defined by F_{se} might lose the intuitive spherical shape and appear distorted. The spherical shape of the cluster is important as it is supported by the notion of spatial *closeness*. If the cluster is distorted e.g., it has an ellipsoidal shape we will have to explain why a particular spatial direction is preferred over another while the choice of the closest object is being made.

If we ignore the distortions and keep the cluster spherical, we preserve the visualization metaphor but we are likely to lose in the performance of the search strategy. To keep the cluster spherical while running the clustering algorithm we use the Euclidean distance between points in the visualization instead of the dissimilarity between the original document vectors. The search strategy function is then defined as

$$F_{se,i}(\mathcal{D}_t, d) = -\frac{1}{|\mathcal{R}_t|} \cdot \sum_{x \in \mathcal{R}_t} dist_i(x, d)$$

where $F_{se,i}(\mathcal{D}_t, d)$ is the search strategy function for the dimension i, $\mathcal{D}_t$ is a representation of the current state of the document set that includes the relevance judgments about the examined documents at time step t, d is an unexamined document, and $dist_i(x, d)$ is the distance or dissimilarity between two document representations in the corresponding space. The index i takes values of "t" for the original vector space, 2 for a two-dimensional embedding, and 3 for a three-dimensional one.

The second question we investigate in this study is how much of the search strategy quality we will sacrifice by preserving the consistency of the visualization. Our intuition is that a higher dimensional visualization will provide more degrees of freedom and therefore it has a better chance to represent the inter-document relationships accurately than a lower dimensional one. We expect that the search strategy in a 3-dimensional visualization will exhibit better performance than in 2 dimensions and the search strategy in the original document-vector space will be the best.

4.1.4 Evaluation Measure

Section 2.3.4 introduces average precision as our evaluation measure in this work. In this chapter we only conduct experiments with the second initial condition: a minimum of relevance information is available at the start of the search. We are interested in how much the search strategies differ in their ability to locate the rest of the relevant material. In the section that follows we present average precision numbers computed for the retrieved document set excluding the documents known at the beginning of the search.

Table 4.1. Performance of the search strategies for the ranked list (RL), interactive relevance feedback (RF), and spring-embedding visualization (F_{se}). Average precision numbers, percent improvement over the simple ranked list are shown. Three different cases of the F_{se} search strategy are considered: one in the original document vector space ($F_{se,t}$), another in 2 dimensional space ($F_{se,2}$), and another in 3 dimensional space ($F_{se,3}$).

Data Set		RL	RF +100t	$F_{se,t}$ (v. RL%)	$F_{se,2}$ (v. RL%)	$F_{se,3}$ (v. RL%)
TREC-5	Full	36.17	44.17	42.44 (17.34 %)	40.71 (12.54 %)	41.62 (15.06 %)
	Desc	34.09	50.60	48.73 (42.95 %)	45.82 (34.43 %)	47.24 (38.58 %)
	Title	30.95	41.39	37.95 (22.61 %)	39.01 (26.04 %)	38.86 (25.54 %)
	ExpTtl	30.95	38.26	37.44 (20.94 %)	36.07 (16.54 %)	37.16 (20.05 %)
TREC-6	Full	51.02	54.70	53.63 (5.12 %)	52.88 (3.64 %)	54.87 (7.53 %)
	Desc	42.48	60.39	57.79 (36.05 %)	52.48 (23.56 %)	55.24 (30.05 %)
	Title	40.10	54.74	55.39 (38.12 %)	52.98 (32.10 %)	54.37 (35.57 %)
	ExpTtl	47.21	50.07	52.50 (11.19 %)	52.38 (10.94 %)	53.52 (13.35 %)
total average		39.12	49.29	48.23 (23.29 %)	46.54 (18.97 %)	47.86 (22.33 %)
titles only		35.53	48.06	46.67 (31.36 %)	45.99 (29.46 %)	46.61 (31.20 %)

4.1.5 Experiments

The first experimental question is to compare the F_{se} strategy with the search strategies for the ranked list and for interactive relevance feedback. Table 4.1 shows the average precision numbers for the F_{se} strategy in the original vector space ($F_{se,t}$), in 2 dimensions ($F_{se,2}$), and in 3 dimensions ($F_{se,3}$). The values for the strategy operating in low dimensions ($F_{se,2}$ and $F_{se,3}$) are averaged across all possible threshold values.

The numbers indicate that $F_{se,t}$ strategy outperforms the ranked list. Most of the differences indicated are statistically significant. We observe a 23.3% increase in precision across different query complexities and size. It is also important to notice that for the "Title" queries – the two or three word queries that are more likely to be entered by hand than the "Full" queries that require a lot of processing – the performance increase is even greater: 31.4%.

The $F_{se,t}$ strategy does slightly worse than the traditional relevance feedback approach (-2.2%). However, that difference is not statistically significant. The difference between $F_{se,t}$ and the less demanding version of the relevance feedback ("RF +10t" in Table 2.4) is also small but positive (4.8%). This method is also enormously more efficient than the relevance feedback approach.

Our search strategy works well in two and three dimensions ($F_{se,2}$ and $F_{se,3}$). In fact, even in two dimensions it is significantly better than the ranked list. It performs worse than the interactive relevance feedback: the average drop in precision is -5.6% and -2.9% accordingly. The observed differences between $F_{se,2}$ and $F_{se,3}$ and the relevance feedback are not significant on the individual data sets. The differences in the total average value, that are computed over all 400 data points, are statistically significant.

4.1.5.1 Effect of Fewer Dimensions

The second question is to compare the performance of the search strategy using the dissimilarities between document vectors against the same strategy that used Euclidean distances in two and three dimensions. The document representations in 2- and 3-dimensional spaces were created using the spring-embedding algorithm. We used the same data set as in the previous experiment.

Table 4.2 shows a small drop in precision when the search strategy is moved from the high-dimensional document space into a smaller number of dimensions. The drop in precision is less when three dimensions are used (-0.8%) instead of two (-3.5%) and it is statistically significant for the total average difference between $F_{se,2}$ and $F_{se,t}$. It is not significant for the difference between $F_{se,3}$ and $F_{se,t}$.

Table 4.2. Performance of the search strategy in the spring-embedding visualization (F_{se}). Three different cases of the F_{se} search strategy are considered: one in the original document vector space ($F_{se,t}$), another in 2 dimensional space ($F_{se,2}$), and another in 3 dimensional space ($F_{se,3}$). Average precision numbers, percent improvement over the same search strategy in the document vector space (the first line in each row), and percent improvement over the same search strategy in 2D (the second line in each row) are shown.

Data Set		F_{se}		
		$F_{se,t}$	$F_{se,2}$ (v. $F_{se,t}$%)	$F_{se,3}$ (v. $F_{se,t}$%) (v. $F_{se,2}$%)
TREC-5	Full	42.44	40.71 (-4.09 %)	41.62 (-1.94 %) (2.24 %)
	Desc	48.73	45.82 (-5.96 %)	47.24 (-3.06 %) (3.08 %)
	Title	37.95	39.01 (2.80 %)	38.86 (2.39 %) (-0.40 %)
	ExpTtl	37.44	36.07 (-3.64 %)	37.16 (-0.74 %) (3.02 %)
TREC-6	Full	53.63	52.88 (-1.40 %)	54.87 (2.30 %) (3.75 %)
	Desc	57.79	52.48 (-9.18 %)	55.24 (-4.41 %) (5.26 %)
	Title	55.39	52.98 (-4.36 %)	54.37 (-1.84 %) (2.63 %)
	ExpTtl	52.50	52.38 (-0.22 %)	53.52 (1.94 %) (2.17 %)
total average		48.23	46.54 (-3.50 %)	47.86 (-0.78 %) (2.83 %)
titles only		46.67	45.99 (-1.45 %)	46.61 (-0.12 %) (1.35 %)

We confirm our intuition about the greater dimensionality of the presentation being more accurate. There is a small improvement (2.8%) if the search strategy is moved from the two-dimensional embedding space to the three-dimensional. This difference is statistically significant only across all 400 data points and is not significant for the individual data sets.

4.1.6 Discussion

These results confirm, in the same way that relevance feedback experiments do, that user feedback can dramatically improve the effectiveness of a ranked list. Unlike most past efforts ([4] being a recent exception) we also show that it is also true when feedback is incremental – and even if no new documents are retrieved. Further, we have confirmed this result in a setting where we believe the user will be able to oversee and control the feedback process.

The visualization that we use to that effect is 2- or 3-dimensional approximation of relationships in a high-dimensional space. Those results show that the dimensionality reduction does not substantially degrade the inter-document similarity information. As one might expect, three dimensional approximation is better than two dimensional since it retains one extra degree of freedom to position the documents. However, past experience has suggested that the extra dimension is of no value to the users [96], perhaps because of additional cognitive overhead. We next investigate that question, along with other user issues.

4.2 User Study

The idea of searching for relevant information by examining the document that is the closest to already discovered relevant material seems simple. We assume that given an accurate visual representation of inter-document similarities the user can effectively locate the relevant documents without any aid from the system.

Thus, the third question of our study is: "How effective in locating the relevant information will the user be when given the spring-embedding visualization of the retrieved set?" We hypothesize that the notion of spatial similarity in the spring-embedding visualization is an intuitive and accurate metaphor for representing inter-document relationships. We expect that the user's search strategy will be similar to ours in both procedure and effectiveness.

To test these hypotheses we have implemented a short computer-based user study. It was designed to simulate a user looking for relevant information in the visualization. Each participant in this study had to solve a number of information-foraging problems. Every problem consisted of a set of white spheres floating in space. The participants were told that the spheres are of two colors: red and green. Initially only one sphere was shown in green and there could also be some red spheres. We only consider the first-relevant initial condition and these spheres correspond to the documents from the first-relevant document subset. The true color of a white sphere could be discovered – the sphere could be "opened up" – by double-clicking on the sphere with the mouse pointer. The participants were asked to find all the green spheres as quickly as possible, trying to avoid opening red spheres. The participants received a small time penalty for opening a sphere – the sphere was animated for several seconds before showing its true color. They were also prohibited from double-clicking on a sphere while another was opening. This was done to discourage the users from mindlessly clicking the spheres in random order. At the same time it crudely simulated the delay that would have been experienced by a person while reading and judging the document.

The participants were told that spheres of the same color (e.g., green spheres) tend to appear in close proximity to each other (similar spheres generally group together) but not necessarily so. The last hint was a direct corollary from the Cluster Hypothesis as the spheres represented the documents, and the color, the document relevance value. A green sphere indicated a relevant document and a red one indicated a non-relevant document. However, the participants were not told the meaning of the spheres. We believe this design eliminates a high uncertainty that is generally connected with query formulation and passing relevance judgments [56, 96] and allows us to isolate the navigation properties of the visualization which are the focus of our study.

The problems were presented in two and three dimensions. The three-dimensional effect was created by using a 3D-rendering engine. To improve the depth perception, a simulated fog effect was added to the picture. The participants were able to rotate, slide, and zoom the set of spheres. The application interface of a two-dimensional presentation was equivalent to that of a three-dimensional one except that the user saw a flat structure on the screen (i.e., he or she could still rotate and zoom the 2-dimensional structure).

Each participant was presented with ten problems. We divided the problems into two equal groups. The problems in one group were shown in two dimensions, the problems in the other – in three dimensions. The dimensions in which each group of problems was shown alternated between users. We also varied the order in which the groups were presented and the order in which the problems inside each group were presented. This was done to account for a possible learning effect. Before each group of problems was shown to a participant he or she was given two training problems to familiarize herself with the application interface. The participants were also asked to fill out questionaries before and after the study.

The study was designed to be completely supervision-free. The software was written in Java and it is available via World Wide Web [107]. We have advertised the study in local newsgroups and in information retrieval mailing lists on the Internet. At the time of this report 40 people have expressed their interest in the study by accessing the software; 20 of them have completed it, spending on average one hour and thirty minutes with the system.

Table 4.3. Users' performance navigating the visualizations of ten randomly selected document sets. The numbers are averaged across all selected document sets. Average precision numbers, percent improvement over the ranked list search strategy, percent improvement over the algorithmic search strategy in the corresponding dimension, and percent improvement of using 3D over 2D are shown. We also show the significance level for each difference by two-tailed t-test.

RL	Algorithm		User			
	2D	3D	2D (v. RL %) (v. Alg 2D%)	significance	3D (v. RL %) (v. Alg 3D%) (v. Usr 2D%)	significance
42.9	59.1	61.4	55.8 (30.1* %) (-5.7* %)	$p < 5 \cdot 10^{-8}$ $p < 5 \cdot 10^{-4}$	53.2 (24.1* %) (-13.3* %) (-4.6* %)	$p < 5 \cdot 10^{-6}$ $p < 5 \cdot 10^{-12}$ $p < 0.01$

4.2.1 Results

To create the problems we randomly selected ten topics from TREC topics 251-350 (see Section 2.4). The following topics were selected: 257, 259, 273, 277, 294, 298, 304, 327, 330, 342. We used the "Title" versions of the corresponding queries to retrieve the 50 top ranked documents for each topic. These documents were visualized and presented to the users in 2 and 3 dimensions. At the beginning of each problem the spheres corresponding to the highest ranked relevant document and the non-relevant documents that precede it in the ranked list were shown in color. The rest of the documents were shown in white – i.e., as if starting after the first relevant document in the ranked list was found. This was supposed to provide the users with the starting point in their exploration.

The users examined the white spheres in sequence. The order in which each user double-clicked the spheres defined the search strategy for that user. To distinguish it from the search strategy discussed in previous experiments (F_{se}) we call the latter the *algorithmic* search strategy. We calculated the average precision for the user's strategy and averaged it across all users and all problems. We observed a significant drop in average precision while comparing the algorithmic search strategy with the users' average performance (Table 4.3). Note, though, that this also indicates that the users do significantly better by using the visualization than by blindly following the ranked list.

The differences between the users' performance in both dimensions and the ranked list, between the users' performance and the algorithmic search strategy in both dimensions, and between the users' performance in individual dimensions are statistically significant by two-tailed t-test with at least $p < 0.01$.

The algorithmic search strategy has a higher performance when working with a 3-dimensional representation of the document set than with a 2-dimensional representation. The users on the other hand show much better results in 2 dimensions than in 3 dimensions. From this observation and also from the comments we have collected during the study we conclude that the users have a much harder time establishing proximity relationships and navigating the 3-dimensional visualization.

We were interested in comparing the users' search strategy with the algorithmic search strategy. Each time the user selected a sphere to examine, the algorithmic search strategy ranks the unexamined (white) spheres by the spatial proximity to the current cluster of examined green (relevant) spheres and assigns a rank number to the user's choice. Note that in this situation the algorithm will select the highest ranked sphere. If both the algorithm and the user select the same sphere, the user's choice is ranked as one. Figure 4.3 shows the rank of the users' choice as each successive sphere is selected. The X-axis is the index of the examined sphere. We show both the average rank and the error bar indicating the standard deviation. We also show the average and standard deviation for the number of green spheres remaining unexamined by the user at each step, and the same number for the algorithm. For example, at the beginning there are 18 unexamined spheres ($x = 0$). In two dimensions both the users and the algorithm succeed in locating a green sphere on first pick – the number of unexamined green spheres drops to 17 for $x = 1$. The users always select

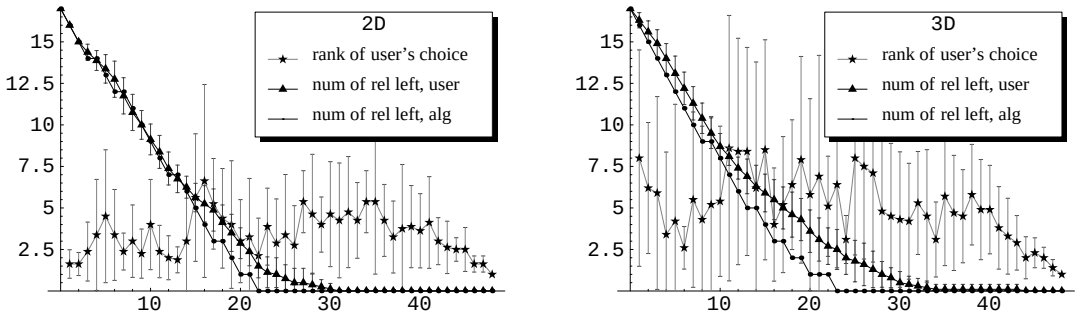

Figure 4.3. Comparison of the users' average search strategy to the algorithmic search strategy in 2 and 3 dimensions for one document set. The X-axis is the number of the examined documents. We show the number of green spheres remaining unexamined by the algorithm ("num of rel left, alg"), the same number for the users ("num of rel left, user"), and the rank of the user choice ("rank of user's choice").

a green sphere – the standard deviation is zero. We also see that the sphere selected by the users on that step has an average rank of 1.5. It means that the users almost always picked up either the first or the second closest white sphere to the first green sphere.

Figure 4.3 presents the algorithm and users' behavior for one of the document sets. We have observed similar effects on the rest of the data. Comparing the plots in 2 and 3 dimensions we see that:

1. the algorithm is more successful in 3 dimensions than it is in 2 dimensions – the plot line for the number of green spheres left after the algorithm's pick descends faster in the right figure (3D).

2. the users are more successful in 2 dimensions than they are in 3 dimensions – the plot line for the number of green spheres left after the users' pick descends faster and trails the same line for the algorithm closer in the left figure (2D).

3. the users' choices have less variance in 2 dimensions than in 3 – the error bars are generally shorter in the left figure (2D).

4. the users' search strategy is more similar to the algorithmic search strategy in 2 dimensions – the average rank of the users' choice is smaller.

5. the users' search strategy is not significantly different from the algorithmic search strategy at least at the beginning of the search – the average users' choice is always less than one standard deviation apart from the algorithm's first choice for the first 14 examined documents.

6. the users' "bad" choices in 2 dimensions – the line for the number of unexamined green spheres left for the users separates upward from the same line for the algorithm in the left figure – correlate with the users' search strategy strongly deviating from the algorithmic search strategy – shown as "picks" on the graph representing the rank of the users' choice at $x = 5$ and $x = 15$.

We asked the users to fill out short questionnaires about their experience with the system comparing 2D with 3D. Specifically, we asked the users to assign a "grade" between 1 and 5 measuring how easy it was to use each presentation and if, in their opinion, the system did a good job at organizing the objects. The average grades in Table 4.4 show that the users preferred the 2D presentation over the 3D one. They overwhelmingly found 2D visualization easier to use and they were generally satisfied with system's arrangement of the green and red spheres.

Table 4.4. Users' responses to a number of questions comparing 2D with 3D visualizations. The answers were given as "grades" between 1 and 5. The average grade is shown for each question. The higher numbers are better ("easier", "more satisfied").

Question	2D	3D
How easy was it to **understand how to use** the system?	4.4	3.4
How easy was it to **learn to use** the system?	4.6	3.6
How easy was it to **use** the system?	4.3	2.7
Are you satisfied with the system's organization of data? Does the system's placement of the objects makes it easier to find the green spheres?	3.4	2.7
Are you satisfied with your performance in finding the green spheres?	3.6	3.1

We conclude that both our hypotheses are supported. The users have no difficulty grasping the idea of spatial proximity as the metaphor for inter-object similarity. Their search strategy is very similar to the notion of selecting the spheres that are close to the known green spheres. In fact, one of the participants explicitly stated he was trying to pick up the "white ball that is the closest to the average of the green set." Generally the users were very successful in following this tactic. However, the more spheres were examined, the more difficult it became to correctly identify the similarity between the cluster of green spheres and the remaining white spheres. This task was even more difficult in three dimensions. The users constantly pointed out that correct identification of the inter-sphere distances in 3D required frequent rotations of the structure, thus making the visualization task more difficult. We believe these effects account for the observed differences in precision between the algorithmic and user search strategies.

4.3 Discussion

In the study conducted by Koenemann and Belkin [56] the users routinely expressed their desire to "see and control" the feedback process. We can explain our algorithm and provide a user with a sense of control by visualizing the search strategy. We present the documents as points in 2- or 3-dimensional space and map the inter-document distances onto the Euclidean distances between points. Then similar documents are shown nearby and the choices of the clustering algorithm are explained: "each time we select the object that is the closest to all relevant documents discovered so far." In fact, it looks like we are growing a spherical cluster from the first relevant document and shifting the center of the sphere as new documents are added to the cluster (see Figure 4.2).

One limitation of this approach is the lack of a clear starting point like the top of the ranked list. That is why we have allowed a minimum of relevance information to be present in the system at the start of the search. We also can consider the query as a document, place it in the visualization and initiate the search strategy from that "query-document" instead of the first relevant document in the ranked list. It will allow us to remove dependency on the ranked list to provide us with the starting point and consider only the clustering visualization. However, a document-query similarity is almost always much lower than a document-document similarity (small queries have few words in common with documents). The query-document will appear as a disc that is far away from the rest of the documents. The positioning error created by the MDS algorithm will be far greater for the query than for the documents and this error might very well result in a bad selection for the seed document.

Our search strategy performs best when there is only one relevant topic in the retrieved set. In that case all the relevant documents form one cluster that is easily detected as soon as the first relevant document is located. The strategy doesn't take into account cases when there are several different but relevant topics and clumps of relevant documents appear to be scattered in the visualization. The users were observed to perform better than the algorithm in such a situation: getting annoyed by discovering many red spheres in one part of the visualization, the users jumped

away and explored a different area. Our search strategy is not able to do that, though it might be possible to provide such an effect.

We have also compared the F_{se} strategy to the F_{cl} from the previous chapter. The differences are small, e.g., $F_{se,t}$ performs better by 3.2%, and not statistically significant. One should note that the F_{se} strategy achieves the same level of performance without using the information about non-relevant documents and the original document ranking.

Our study showed that the user while being significantly more successful with the spring-embedding visualization than with the ranked list might require some aid in pinpointing the neighborhood spheres. This suggests that there are some usability issues whose solution may improve the overall system. For example, it was suggested that the system interface should allow a user to adjust the radius of spheres. Increasing the size immediately reveals the closest pairs of spheres – those spheres begin to touch or intersect each other. We do not study the usability question in this thesis. Instead we focus on how the system can provide a more direct assistance to the user, for example by highlighting the most likely relevant document. The document organization can supply such help by employing the services of the relevance feedback approach based on query expansion. In the next chapter we consider an alternative to the traditional relevance feedback that extends our ideas for the F_{se} strategy and accommodates additional information about the document sets. In Chapter 6 we will describe how such an assistance can be implemented in a real-world system.

4.4 Related Work

Multiple visualization approaches for information organization have been developed in recent years. Generally these visualizations are designed to present some type of patterns in a document set and are considered to be browsing interfaces. The documents are usually presented as points or objects in space with their relative positions indicating how closely they are related. Links are often drawn between highly-related documents to make the fact that there is a relationship clearer. To our knowledge there have been no studies on how such visualizations help the user locate relevant information.

Pejtersen [78] suggests a BookHouse metaphor for an online library catalog. Icons and pictures represent different search strategies, databases, rooms, contexts. For example, if there is a picture with only children in the room – it is books for children database. There are 108 icons to represent 1000 terms and concepts. The author reports a big satisfaction with the retrieved results.

Rose et al. [82] suggest a replacement for the well-known folder metaphor: *pile*. A pile is a collection of documents that are brought together according to an arbitrary complex rule attached to the pile. In essence this metaphor presents an interesting interface to an automatic classification system. The system supports automatic filing, overlapping piles, scanning through a pile, automatic keyword assignment to piles.

4.4.1 2-D Visualization

Croft and Thompson designed an information retrieval system I^3R [30] that displays retrieved documents in a network based on inter-document similarity. Other systems [9, 71] present the inter-document similarity hierarchically.

Allan [3, 5] developed a visualization for showing the relationship between documents and parts of documents. It arrayed the documents around an oval and connected them when their similarity was strong enough. Allan's immediate goal was not to find the groups of relevant documents, but to find unusual patterns of relationships between documents.

The Vibe system [34] is a 2-D display that shows how documents related to each other in terms of user-selected dimensions. The documents being browsed are placed in the center of a circle. The user can locate any number of terms along the edge of the circle, where they form "gravity wells" that attract documents depending on the significance of that terms in that document. The user can shift the location of terms and adjust their weights to better understand the relationships between the documents.

Some systems employ relationship lattices to isolate the conceptual structure in the collection [41] or to provide a personal thesaurus space [77]. The latter is used for query formulation. If such a representation were to be applied to retrieval results, the layout would be too complex to read.

In the system designed by Lin [69], a Kohonen's Feature Map algorithm creates maps that characterize the overall content of a document collection. The regions of the map are defined by single words or phrases and they may vary in size and shape corresponding to the frequency the word (or phrase) occurs in the collection.

4.4.2 3-D Visualization

High-powered graphics workstations and the visual appeal of 3-dimensional graphics have encouraged efforts to present document relationships in 3-space. The LyberWorld system [51] includes an implementation of the Vibe system described above, but presented in 3-space. The user still must select terms, but now the terms are placed on the surface of a sphere rather than the edge of a circle. The additional dimension should allow the user to see separation more readily.

Galaxies [105] computes word similarities and displays the documents as a universe of "docustars". In a space with a huge number of "docustars" it is not easy to select the object the user wants to explore.

Other approaches exist that attempt to visualize the document space based on the concept similarities using some form of neural network such as Kohonen's self-organizing maps [69, 83]. The Narcissus system [52] applies the spring-embedding algorithm to visualizing structures of pages and links on the World Wide Web. Song experimented with visualization of clusters for a bibliographic database [92].

Several systems place the documents as points in space. The users are supposed to detect classes of similar document in the arrangements of points. These systems include ThemeScapes [105], the Galaxy of News [79] and Bead [24].

Bead [24] views documents as multidimensional vectors of terms. It uses a the spring-embedding algorithm to visualize the document set as points in 3-dimensional space. The Bead research was done on a small collection of abstracts. When attempting to visualize complete documents they faced a threshold selection problem – they had to decide where to draw a boundary between similar and dissimilar documents. Without making such a decision the visualization tends to completely lose any meaningful structure. They did not solve that problem. That study did not investigate the question of separating relevant and non-relevant documents.

Swan and Allan [96] considered a system with the ranked list and the spring-embedding visualization. Their system presented complete, full-sized documents and it was studied in the context of searching for multiple topics across several query runs. There was no exploration of how the ranked list and the visualization could be effectively used together.

4.5 Summary

In this chapter we present a visualization approach that arranges the retrieved documents based on their similarity. We show that it can be used effectively to interactively direct the user's search for relevant information in the top ranked portion of the retrieved set. We have experimentally shown that this approach significantly exceeds the initial performance of the ranked list and rivals in its effectiveness the traditional relevance feedback methods.

We argue that an approach such as this one is easier to understand because it does not involve rearranging documents nor changing the visualization, giving the user a sense of control and reducing potential confusion. We have shown that the retrieved set can be visualized in 2- and 3-dimensions and the same procedure of selecting the documents based on their proximity to the relevant material can be repeated in the fewer dimensions with just an insignificant loss in precision.

Visualizing the data in fewer dimensions decreases the accuracy of the presentation. We observed that a three-dimensional visualization is more accurate in representing the inter-document relationships than a two-dimensional one. Consequently, the document selection algorithm is more effective in three dimensions than in two.

We also showed that spatial proximity is an intuitive and well-recognized (by the users) metaphor for similarity between objects. We observed that the users' search strategy tends to follow the model incorporated into our algorithmic approach. The users' were significantly more successful with the visualization than they would be by following the ranked list. The precision difference between the users and the algorithm arises from difficulty to precisely identify the inter-object distances in the visualization. We also observed that the higher accuracy with which 3-dimensional structure represents the document set is negated by the heavy cognitive effort that is required from the users to navigate the visualization.

CHAPTER 5

WIZARD

In Chapter 3 we analyzed an information organization system based on a clustering algorithm. We designed a search strategy that a user can follow to locate relevant documents quickly. That strategy was effective but simple and hand-crafted. How can we craft a search strategy automatically?

Chapter 4 describes a document organization that uses the spring-embedding algorithm to visualize the documents as spheres in two and three dimensions. We found that such a representation is very powerful – good performance can be achieved by following a simple rule. However, unassisted searchers are unable to use the system to its full potential. We want to provide users with an automatic artificial "adviser" or "wizard" that will help them to navigate the system and highlight the most likely relevant documents. How effective can we make such a wizard?

These two questions have one single answer. Indeed, if we can create a search strategy automatically, we can make sure it is effective. That strategy forms the core of the wizard: the document organization system collects feedback from the user, computes the search strategy function to estimate the document relevances, and points out the best documents to the user.

The search strategies we consider in this dissertation use only the informational clues directly supplied by the document organization system. This condition ensures that the strategy can always be explained by the document presentation part of the system. Then the question of creating an effective search strategy becomes the question of combining the information clues in the most effective way. In contrast to this approach, the interactive relevance feedback based on query expansion is a poor candidate for the wizard tool. It incorporates data about the individual terms and original query which is not represented in the systems we analyze in this thesis.

In this chapter we consider three different representations of a search strategy for the spring-embedding visualization and show that a learning algorithm called reinforcement learning can be successfully applied to create very effective approaches that surpass both the ranked list and traditional relevance feedback. We conduct our analysis following the SBE framework.

5.1 Reinforcement Learning

The problem of navigating the retrieved document set can be naturally expressed as a reinforcement learning problem [95]. Indeed, we have a search strategy operating in a document set. As we described in Section 2.3.3 the search strategy interacts with the document set at discrete time steps. At each time step t the environment state is defined by the inter-document similarities, what documents were examined, and what relevance judgments were assigned. The search strategy receives some representation of the environment state ($\mathcal{D}_t$) and computes a numerical value of the search strategy function for each unexamined document d: $F(\mathcal{D}_t, d)$. In the previous chapters we used the $F(\mathcal{D}_t, d)$ values to order the documents and we measured the quality of that ordering.

Let us extend the definition of the search strategy and allow it some autonomy. We will view the search strategy as an agent operating in an environment (i.e., our document set). The agent has to select an action – choose the next document d to examine. At each time step it chooses the document with the highest $F(\mathcal{D}_t, d)$. At the next time step the agent receives a numerical reward from the environment – whether the examined document is relevant or not. The interaction process continues until all documents are examined. The agent's goal is to maximize the total reward – find all the relevant documents as quickly as possible. This is a precise formulation of a reinforcement learning problem [95, p.51]. Reinforcement learning methods specify how the search strategy function can be changed as a result of the interaction experience with the goal of maximizing the total reward.

initialize $\vec{\theta}$ arbitrarily
$\mathcal{D} \leftarrow$ initial state of document set
$d \leftarrow \arg\max_{\forall x \in \mathcal{U}} F_{\vec{\theta}}(\mathcal{D}, x)$
with probability ϵ: $d \leftarrow$ a random document in $\mathcal{U}$

repeat while $\mathcal{U} \neq \emptyset$:
    examine document d
    observe reward r, and next document set configuration, $\mathcal{D}'$
    $d' \leftarrow \arg\max_{\forall x \in \mathcal{U}'} F_{\vec{\theta}}(\mathcal{D}', x)$
    with probability ϵ: $d' \leftarrow$ a random document in $\mathcal{U}'$
    $\delta \leftarrow r + \rho F_{\vec{\theta}}(\mathcal{D}', d') - F_{\vec{\theta}}(\mathcal{D}, d)$
    $\vec{e} \leftarrow \nabla_{\vec{\theta}} F_{\vec{\theta}}(\mathcal{D}, d)$
    $\vec{\theta} \leftarrow \vec{\theta} + \eta \delta \vec{e}$
    $\mathcal{D} \leftarrow \mathcal{D}'$
    $d \leftarrow d'$
end repeat

Figure 5.1. Temporal Difference learning algorithm adapted for training a search strategy function.

We focus on a particular reinforcement learning algorithm called Temporal Difference (TD) learning [95]. The particular property of this algorithm is that it adjusts the agent (F) using the difference in the agent's evaluation of the document set before and after an action is taken. Figure 5.1 shows the TD algorithm adapted for a search strategy function. There $\mathcal{U} \subset \mathcal{D}$ denotes the set of all unexamined documents, $\vec{\theta}$ is the parameter vector that define the search strategy function $F_{\vec{\theta}}(\mathcal{D}, d)$, η is the learning rate that controls the speed and stability of the learning process, and ρ is the discount factor that controls how much the reward at time $t+1$ is discounted by comparing to reward at time t.

The search strategy strongly depends upon the reward function r. In this thesis our reward function $r_{t+1} = 1$ if the examined document d_t was relevant and 0 otherwise – we reward the search strategy for discovering the relevant documents. However, the total reward will not depend upon when the relevant documents are examined, i.e., the total reward will be the same if all relevant documents are examined at the beginning of the search or at the end. The discount factor ρ is a part of the reward function that decreases the reward if it was obtained at a later time step. If $\rho < 1$, the search strategy will prefer the relevant documents to occur towards the beginning of the search. At the same time it can allow for some exploration of the document set if the information collected during the exploration will help to find the relevant documents in one quick sweep. If $\rho = 0$, the search strategy will prefer immediate rewards and it will not do any exploration.

5.2 Experimental Task

Section 2.3.1 defines two initial conditions for the experiments – the no-information and first-relevant initial conditions. In contrast to Chapter 4 where we focused solely on the second condition, i.e., minimum relevant information is available at the beginning of the search, we study both initial conditions.

5.3 System Design

The document organization system integrates the spring-embedding visualization of documents with the ranked list. Figure 5.2 illustrates how such system can be implemented. The spring-

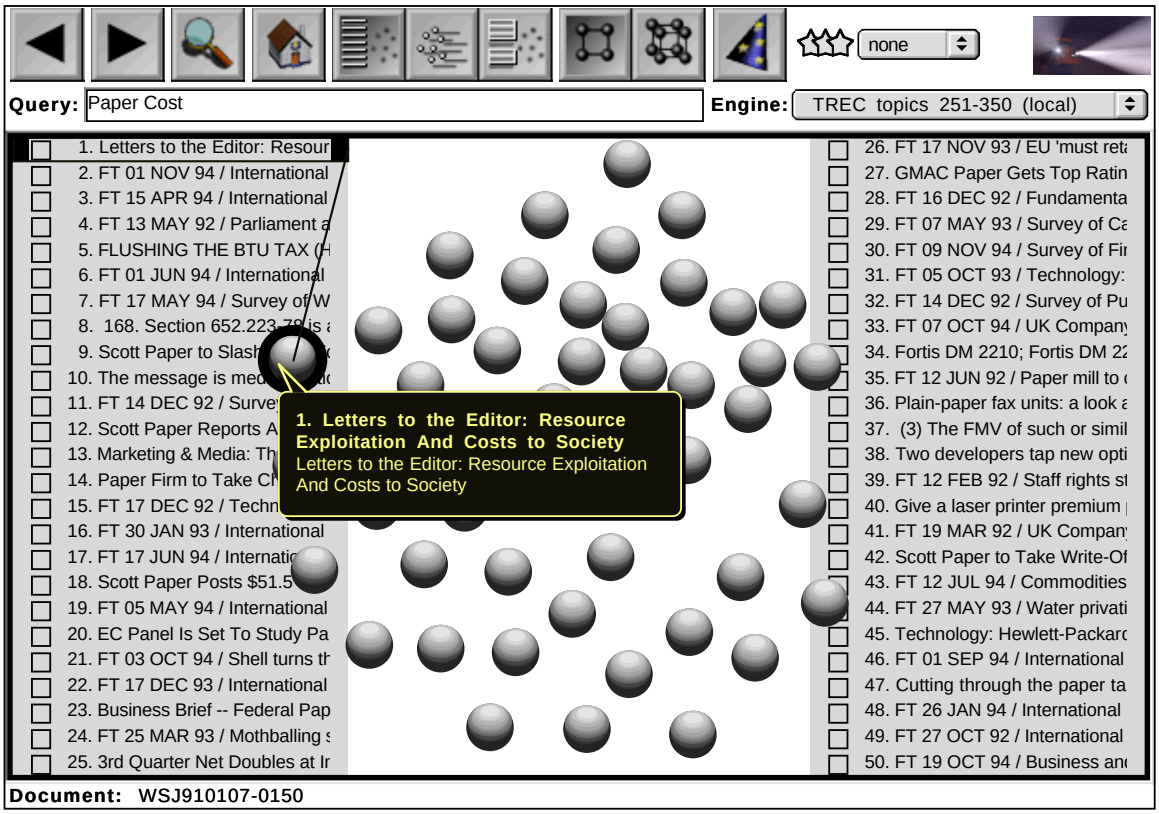

Figure 5.2. Screen shot of the system that integrates the spring-embedding visualization and the ranked list. The top fifty documents retrieved by the INQUERY search engine for the "Paper Cost" query. A two-dimensional picture is shown.

embedding visualization occupies the center of the pictures with the ranked list placed in two columns around it. Chapter 6 provides a more detailed description of the implementation.

5.4 Search Strategy

In the next three sections we consider the question of selecting good features to represent the document set state and define three different forms for the search strategy function. Our analysis is based on the ideas from the Rocchio feedback approach, where the document ranking is adjusted based on the information from the old query and examined documents.

5.4.1 Simple Rocchio

The first design follows directly from the Rocchio approach [81] (see also Section 2.6.6). We define the search strategy function $F_{Roc}(\mathcal{D}, d)$ as a weighted sum that has four arguments: a constant, document similarity to the query, average similarity between the document and all examined relevant documents, and average similarity between the document and all examined non-relevant documents:

$$\begin{aligned} F_{Roc}(\mathcal{D}_t, d) &= \theta_0 + \theta_1 \cdot querysim(d) \\ &+ \theta_2 \cdot \frac{1}{|\mathcal{R}_t|} \sum_{x \in \mathcal{R}_t} sim(x, d) \end{aligned}$$

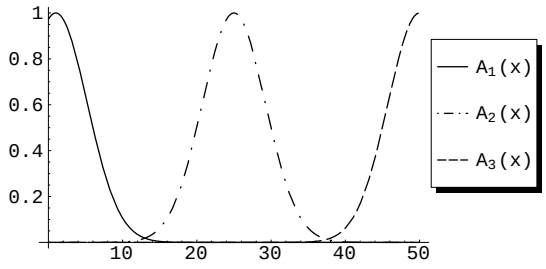

Figure 5.3. The application coefficients used for training $F_{AC}(\mathcal{D}_t, d)$. $|\mathcal{D}| = 50$.

$$+ \quad \theta_3 \cdot \frac{1}{|\mathcal{N}_t|} \sum_{x \in \mathcal{N}_t} sim(x, d)$$

where $querysim(d)$ is the similarity between the document and the original query, $sim(x, d)$ is the similarity between two documents, $\mathcal{R}_t$ and $\mathcal{N}_t$ are the set of all examined relevant and non-relevant documents at time step t.

Note that the F_{se} search strategy from the previous chapter is equivalent to the F_{Roc} strategy with $\theta_0 = \theta_1 = \theta_3 = 0$ and $\theta_2 = 1$.

5.4.2 Application Coefficients

Our second design results from an heuristic observation that different arguments in the Rocchio-based representation should be weighted depending on how many documents were examined. For example, the similarity of a document to the query is more important at the beginning of the search, when few relevant documents are known, than further into the process when the document relevance information is plentiful. We accommodate this intuition by dividing the search process into three distinct phases and training a separate search strategy function for each phase. Specifically, we define our second search strategy function $F_{AC}(\mathcal{D}, d)$ as a linear combination of three instances of the search strategy function from the previous section ($F_{Roc}(\mathcal{D}_t, d)$), where the coefficients – called *application coefficients* in machine learning – are smooth functions of the number of the examined documents [16]. This approach has been shown to be successful in developing neural networks for game playing [60, 65].

$$F_{AC}(\mathcal{D}_t, d) = \sum_i A_i(|\mathcal{R}_t| + |\mathcal{N}_t|) \cdot F_{Roc,i}(\mathcal{D}_t, d)$$

Here $A_i(\cdot)$ is defined as follows:

$$A_i(x) = \exp\left(-\frac{(x - \mu_i)^2}{\sigma^2}\right)$$

where μ_i defines the center of the influence area for the ith subnetwork $F_{Roc,i}(\mathcal{D}_t, d)$ and σ defines its width (Figure 5.3).

5.4.3 Tile Coding

Both search strategy function representations discussed in the previous sections are limited in the shape of the functions they can approximate: $F_{Roc}(\mathcal{D}, d)$ describes a linear combination of features or a hyperplane, while $F_{AC}(\mathcal{D}, d)$ describes three connected hyperplanes. For our third representation we used a technique called *tile coding* that can approximate more complex functions [95, p. 204].

In tile coding the feature space is partitioned with a regular grid (*tiling*) and a single number is assigned to each cell (*tile*) in the partition. Generally, several tilings are placed in the feature space

and their origins are shifted by some amount relative to each other. The set of tilings defines the final function $F_{Tile}(\mathcal{D}, d)$: given a point in the feature space, a tile containing that point is selected from each grid and the average of the corresponding numbers is returned. Detailed overview of the tile coding and a good public-domain implementation of this technique can be found elsewhere [27].

The flexibility of tile coding comes at a cost of losing some accuracy while partitioning the feature space. The accuracy of the representation can be improved by increasing the number of tilings and decreasing the size of individual tiles. The generalization ability of the approximation is proportional to the tile size as well. Requiring the large number of tilings makes the tile coding much more computationally expensive than the simple linear approximator.

The search strategy function $F_{Tile}(\mathcal{D}, d)$ is defined on the feature space of five dimensions. Four of them are the same features that are used in $F_{AC}(\mathcal{D}, d)$: document-query similarity, average similarities to relevant and non-relevant documents, and the number of examined documents. The fifth feature is the number of examined documents squared. This feature was added after we observed in our preliminary experiments that $\vec{\theta}$, as a function of the number of examined documents learned by $F_{AC}(\mathcal{D}, d)$, exhibited a non-linear form.

5.5 Evaluation Measure

Section 2.3.4 introduces average precision as our evaluation measure. In this chapter we conduct experiments with both initial conditions and compute average precision for the whole document set (with the no-information condition) and for unexamined portion of the document set (with the first-relevant starting condition).

5.6 Experiments

We use the eight experimental data sets described in Section 2.4. Each data set contains 50 document sets of 50 documents each. The second initial condition was in effect: each document set had the first relevant document and all preceding non-relevant documents marked with the corresponding relevance judgments. We divided the eight data sets – one for each query type on each collection – into three subsets: training, testing, and evaluation. The document sets retrieved from different collections do not overlap. The document sets retrieved from the same collection exhibit some amount of overlapping. For example, each of four queries of different types created for the same TREC topic retrieve fifty documents creating a set of two hundred documents total. Our analysis shows that there are on average a hundred unique documents in that set. We trained each search strategy function on one data set, making it four functions trained on the data sets from one collection. We selected one function out of these four that performed best on the data from that collection overall. We then evaluated that function on the four data sets retrieved from another collection.

We trained the search strategies by running them on the training set multiple times. Each search strategy function began with all parameters ($\vec{\theta}$) initialized to zero. We have experimented with different values for the TD-algorithm parameters. The learning rate $\eta = 0.1$ and discount factor $\rho = 0.4$ worked well in our experiments. The parameters for application coefficients ($A(\cdot)$) were defined as $\mu = 1, 25, 50$ and $\sigma = 6$. This divides the process into three distinct phases: the beginning phase of the search that spans over the first 12 examined documents, the middle phase for the next 24-25 documents, and the end phase for the last 12 documents. The σ parameter ensures that the behavior of the $F_{AC}(\mathcal{D}, d)$ strategy on each phase is controlled primarily by the corresponding $F_{Roc,i}(\mathcal{D}_t, d)$ [65]. The tile coding representation used 256 tilings, each tile side was equal to one third of the corresponding feature range. The learning process was terminated at the point when the average precision failed to improve for several iterations.

Table 5.1. Average precision and percent improvement over the ranked list (in parentheses) for the top fifty retrieved documents. The second initial condition is in effect: the search strategies are given the relevance judgments for the top ranked relevant document and all non-relevant documents that precede it in the ranked list. Precision is computed for the unexamined portion of the retrieved set.

Data Set		RL	RF +100t	$F_{Roc}(\mathcal{D}, d)$ (v. RL%)	$F_{AC}(\mathcal{D}, d)$ (v. RL%)	$F_{Tile}(\mathcal{D}, d)$ (v. RL%)
TREC-5	Full	36.17	44.17	46.55 (28.70 %)	50.65 (40.03 %)	51.00 (41.00 %)
	Desc	34.09	50.60	52.96 (55.37 %)	53.00 (55.49 %)	53.42 (56.72 %)
	Title	30.95	41.39	41.96 (35.57 %)	43.41 (40.25 %)	43.99 (42.13 %)
	ExpTtl	30.95	38.26	40.73 (31.59 %)	43.76 (41.38 %)	44.02 (42.22 %)
TREC-6	Full	51.02	54.70	57.85 (13.38 %)	61.84 (21.20 %)	62.38 (22.26 %)
	Desc	42.48	60.39	60.52 (42.48 %)	62.85 (47.96 %)	63.29 (49.00 %)
	Title	40.10	54.74	57.26 (42.78 %)	58.36 (45.52 %)	58.95 (46.99 %)
	ExpTtl	47.21	50.07	55.38 (17.30 %)	56.66 (20.01 %)	57.03 (20.79 %)
total average		39.12	49.29	51.65 (32.03 %)	53.82 (37.56 %)	54.26 (38.69 %)
titles only		35.53	48.06	49.61 (39.64 %)	50.89 (43.23 %)	51.47 (44.87 %)

5.7 Results

Tables 5.1, 5.2, and 5.3 show the average precision numbers obtained for the ranked list, interactive relevance feedback, and our three search strategies (F_{Roc}, F_{AC}, and F_{Tile}) and the percent improvement over the ranked list.

Table 5.1 shows the average precision values for experiments with the second starting conditions. This experiment is equivalent to the one in Chapter 4 (see Table 4.1). We observe a significant improvement in the average precision for all search strategies over the ranked list. The performance increases as more information is added to the document set representation. The first search strategy learns a set of Rocchio coefficients and exhibits a 32.0% improvement over the baseline. The second strategy allows these coefficient to change as the browsing process progresses and achieves a 37.7% improvement. The third search strategy that uses tiling to represent its function demonstrates a 38.7% improvement.

The performance increase is even greater if only the "Title" queries are considered (i.e., 39.6%, 43.2%, and 44.9%).

All three search strategies outperform the interactive relevance feedback search strategy. The total average differences are 4.8%, 9.2%, and 10.1% respectively. These improvements are statistically significant.

We observed a similar increase in average precision while evaluating the search strategies on the document sets containing the top 100 documents instead of the top 50 (Table 5.2). The differences between the F_{Roc}, F_{AC}, and F_{Tile} strategies and the interactive relevance feedback strategy are 7.0%, 11.2%, and 12.6% respectively. These improvements are statistically significant.

We repeated the evaluation on the same document sets with the first initial condition: no relevant information was available to the strategies at the beginning of the search. The document sets of top 50 documents were used in this experiment. Table 5.3 shows the average precision numbers computed in those settings. We again observe a statistically significant improvement in average precision over both baseline system, albeit a smaller one. The differences between the F_{Roc}, F_{AC}, and F_{Tile} strategies and the interactive relevance feedback strategy are 6.4%, 10.8%, and 11.4% correspondingly. These differences are statistically significant.

5.7.1 Interpretation

The first two search strategy functions $F_{Roc}(\mathcal{D}, d)$ and $F_{AC}(\mathcal{D}, d)$ are easily interpreted by considering the weight coefficients learned in the training process. The size (256 tilings and more than

Table 5.2. Average precision and percent improvement over the ranked list (in parentheses) for the top one hundred retrieved documents. The second initial condition is in effect: the search strategies are given the relevance judgments for the top ranked relevant document and all non-relevant documents that precede it in the ranked list. Precision is computed for the unexamined portion of the retrieved set.

Data Set		RL	RF +100t	$F_{Roc}(\mathcal{D}, d)$ (v. RL%)	$F_{AC}(\mathcal{D}, d)$ (v. RL%)	$F_{Tile}(\mathcal{D}, d)$ (v. RL%)
TREC-5	Full	33.36	44.68	45.81 (37.31 %)	49.86 (49.45 %)	50.59 (51.64 %)
	Desc	30.37	47.87	48.37 (59.25 %)	48.52 (59.74 %)	48.92 (61.06 %)
	Title	26.78	40.44	42.03 (56.94 %)	43.14 (61.09 %)	44.00 (64.30 %)
	ExpTtl	28.51	35.18	37.82 (32.67 %)	39.94 (40.11 %)	40.80 (43.12 %)
TREC-6	Full	45.16	50.68	55.61 (23.14 %)	60.01 (32.88 %)	60.29 (33.50 %)
	Desc	38.82	58.05	61.69 (58.90 %)	64.41 (65.91 %)	64.74 (66.76 %)
	Title	36.21	51.21	55.71 (53.87 %)	55.20 (52.46 %)	56.48 (56.00 %)
	ExpTtl	43.01	46.71	53.86 (25.24 %)	55.79 (29.73 %)	56.22 (30.73 %)
total average		35.28	46.85	50.11 (42.05 %)	52.11 (47.71 %)	52.76 (49.54 %)
titles only		31.49	45.82	48.87 (55.18 %)	49.17 (56.13 %)	50.24 (59.53 %)

Table 5.3. Average precision and percent improvement over the baseline (in parentheses) for the top fifty retrieved documents. The first initial condition is in effect: the search strategies start without any relevance information: precision is computed for the whole data set.

Data Set		RL	RF +100t	$F_{Roc}(\mathcal{D}, d)$ (v. RL%)	$F_{AC}(\mathcal{D}, d)$ (v. RL%)	$F_{Tile}(\mathcal{D}, d)$ (v. RL%)
TREC-5	Full	38.49	41.87	46.06 (19.67 %)	47.58 (23.62 %)	48.19 (25.21 %)
	Desc	36.06	44.19	44.65 (23.83 %)	45.77 (26.94 %)	47.27 (31.10 %)
	Title	33.19	36.29	42.01 (26.59 %)	42.88 (29.21 %)	43.37 (30.69 %)
	ExpTtl	29.91	32.64	36.21 (21.05 %)	40.01 (33.76 %)	40.32 (34.79 %)
TREC-6	Full	53.67	54.48	57.15 (6.47 %)	62.44 (16.33 %)	60.46 (12.64 %)
	Desc	46.66	58.54	58.95 (26.33 %)	60.09 (28.78 %)	60.45 (29.55 %)
	Title	46.19	56.23	57.95 (25.47 %)	59.19 (28.15 %)	60.24 (30.42 %)
	ExpTtl	51.64	53.12	58.37 (13.04 %)	60.09 (16.37 %)	60.16 (16.50 %)
total average		41.98	47.17	50.17 (19.52 %)	52.26 (24.49 %)	52.56 (25.21 %)
titles only		39.69	46.26	49.98 (25.94 %)	51.04 (28.60 %)	51.81 (30.54 %)

Table 5.4. Rocchio coefficients from three different sources.

System	Old query (θ_1)	Relevant (θ_2)	Non-rel. (θ_3)
INQUERY [6]	0.5	4	-1
DFO [20]	0.25	8	-1
$F_{Roc}(\mathcal{D}, d)$	0.5	2	-1

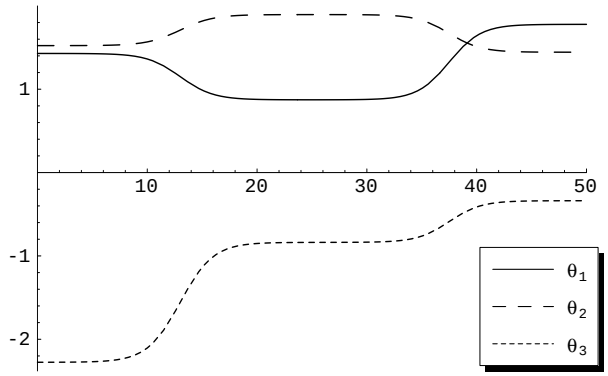

Figure 5.4. The Rocchio coefficients vary significantly with the number of examined documents.

20000 tiles) and complexity (5 dimensions) of the tile coding representation make the interpretation of the final $F_{Tile}(\mathcal{D}, d)$ function very difficult and we do not attempt it.

The first representation – the straightforward adaptation of the Rocchio approach – showed values that are significantly different from those discussed in the past studies (Table 5.4). INQUERY relevance feedback subsystem uses the relevant to non-relevant ratio of $4:1$ and it has been shown to work well in TREC experiments [6]. We have considered a variant of $F_{Roc}(\mathcal{D}, d)$ with INQUERY's relevant-to-non-relevant weight ratio. It exhibited performance very similar to what we have observed from the interactive relevance feedback search strategy. Buckley and Salton [20] recommend the ratio of $8:1$. We implemented another version of $F_{Roc}(\mathcal{D}, d)$ that used this ratio and observed a slight but not statistically significant drop in average precision while comparing it to the interactive relevance feedback search strategy. In our experiments the search strategy function consistently learned the same $2:1$ ratio for the Rocchio coefficients on each individual training set.

The second search strategy function representation was allowed to change the Rocchio coefficients depending on how many documents were examined. Figure 5.4 shows the learned coefficients vary with each examined document. We observed that the magnitude of the coefficient for the average similarity to non-relevant documents ($|\theta_3|$) steadily decreases as more documents are examined. The magnitude of the coefficient for the average similarity to relevant documents ($|\theta_2|$) displays an increase for the middle part of the process, while the importance of the document-query similarity ($|\theta_1|$) shows exactly the opposite behavior. Similar graphs were observed for the functions trained on different data sets.

5.8 Discussion

The learning phase for the agent stops when it is finished with the training data set. We have experimented with an idea of allowing the agent to learn while running it on the evaluation set. Specifically, at each time step during the evaluation phase the agent was allowed to learn for several iterations using only the examined subset of the data. After each query the agent parameters were restored to their initial (learned on the training set) values. We observed that, on average, this

learning did not have any effect: we saw an increase in average precision for some queries and a degradation for others.

There are several key points that made the reinforcement learning and TD-learning in particular a good choice for our problem. Firstly, the search strategy is learned from experience by interacting with the training examples. That interaction has an intuitive connection to how a user would interact with the system.

Secondly, exhaustive knowledge about the problem space is not required. The TD-learning method allows the search strategy to be learned from samples. Several theoretical results guarantee convergence of the learning process to an optimal search strategy that maximizes the expected total reward. These results are based on the assumption that the problem satisfies the Markov property, i.e., the reward r obtained by the agent after examining a document d at time t depends only on that document and the document set representation at t. This reward does not depend upon document set representation or the document selected at any time $t' < t$ [95]. We did not study if and by how much the problem in our formulation violates the Markov property. However, our experimental results show that convergence does occur and we leave the question about the Markov property for future work.

Lastly, the learning process is directed by means of providing rewards and punishments for each choice made by the search strategy. Such a framework is very flexible – different reward schemes will result in the search strategies maximizing different utility functions. It also requires a careful consideration while selecting a particular reward function. An intuitive choice of using the relevance value as the reward number requires the search strategy to maximize "total discounted relevance" (TDR):

$$TDR(\rho) = \sum_{d_t \in \mathcal{R}} \rho^t$$

while we evaluate the performance using the average precision (P):

$$P = \frac{1}{|\mathcal{R}|} \sum_{d_t \in \mathcal{R}} \frac{|\mathcal{R}_t|}{|\mathcal{R}_t| + |\mathcal{N}_t|}$$

where $\mathcal{R}$ is the subset of all relevant documents in the data set and d_t is the document examined at time step t. Notice that $|\mathcal{R}_t| + |\mathcal{N}_t| = t$.

Although these are two different measures, the resulting search strategies exhibit high values of average precision in our experiments. That can probably be explained by a high correlation between $TDR(\rho)$ and P for low values of ρ. An alternative would be to use P as the reward at the end of the search ($r_{|\mathcal{D}|+1}$) with all intermediate rewards set to zero. Then the search strategy should learn to maximize the average precision. However, we have observed that using such a reward function results in a very slow learning process and a very similar search strategy.

5.9 Related Work

Relevance feedback is an important topic in information retrieval. We reviewed general work on relevance feedback in Section 2.6.6. Here we focus on connectionist approaches.

Neural networks have found successful application in both retrieval and relevance feedback [13, 103, 57, 17]. The network is constructed connecting the documents, terms, and query. The connection weights are initialized using the statistical information about the term usage and adjusted as the relevance judgments become available. In our study we consider only the similarity information between the documents and ignore the data about the individual terms.

Text classification and categorization is where the neural networks are trained to assign class labels to the documents based on the terms they contain [74, 87]. Information filtering also benefits from the neural network classification abilities. Here they are used to select relevant material from document streaming through the system. Schütze et al. [88] compared neural network-based classifiers with other classification techniques. Lewis et al. [67] and later Callan [22] studied different learning methods for the neural networks. Hull et al. [53] examined performance of combined

methods where the decisions of several different classifiers including neural network were considered together.

The growth of the world wide web resulted in the development of autonomous agents that use the hyperlink environment to perform distributed search tasks on behalf of the user. Quite often a neural network forms a core part of such an agent. Menczer and Belew [72] studied the features of the web structure that can be most useful for the agents. The Wisconsin Adaptive Web Assistant [90] compiles its search instructions into a neural network allowing for adjusting of the search strategies as the training information become available. Joachims et al. [55] developed a system that performs look-ahead searches and provide the suggestions to the user on the base of reinforcement learning. Balabanovic [11] studied the agents that make use of preference information collected from multiple users. Choi and Yoo [25] considered how the multiple agents accepting user's feedback, can interact with each other and learn from their common experience. These systems make extensive use of the hyperlink structure and term level information. They slowly adapt to the user and her requests while performing the search task. We are looking at navigating unstructured text documents, training the agent off-line, and keep it unchanged during the actual search session.

5.10 Summary

We modeled the feedback process with an agent that is interacting with the retrieved set. We formalized the agent description and formulated the problem in terms of reinforcement learning. We showed how the agent performance can be improved after a modest amount of training.

We compared three different approaches to the search strategy function and discovered that taking into account the number of examined documents improves our results. We demonstrated that the technique is very successful when only the inter-document similarity data is available and no term information is provided. Thus a very efficient implementation of the feedback algorithm can be easily constructed, making this approach a good candidate for on-line search environments.

Our approach outperforms the traditional interactive relevance feedback based on query expansion by a statistically significant margin. Additionally, this simple and effective procedure is easily interpreted and can be clearly visualized in the spring-embedding system. This creates a search environment where the user can understand and control the feedback process.

CHAPTER 6

LIGHTHOUSE

In Chapter 3 we discussed a system that arranges the documents in a list using a clustering algorithm. In Chapter 4 we considered a document organization system that uses the spring-embedding technique to visualize the documents as spheres floating in space. In Chapter 5 we developed an effective technique that allows us to locate relevant document quickly in a system that integrates the ranked list and spring-embedding visualization.

We have developed a document organization system that integrates all these parts: the ranked list, cluster partitioning of the document set, spring-embedding visualization, and wizard tools developed with reinforcement learning. We call this system Lighthouse.

This chapter describes different features of the Lighthouse system to illustrate how the technologies studied in this thesis can be applied in a real-world on-line search. However, we do not evaluate the system in this thesis, leaving this question for future work.

6.1 Integration of Ranked List and Spring-Embedding

Lighthouse is an interface system. It accepts a query from a user, passes the query to a search engine, receives back a ranked list of documents, arranges the documents and presents them to the user. Lighthouse does not do any document retrieval itself. This allows for a great flexibility in setting up the system. Almost any search engine can be become a backend for Lighthouse. We will present examples of document sets retrieved by INQUERY and AltaVista (`www.av.com`).

Figure 6.1 shows a screen shot of the Lighthouse system. INQUERY ran the query "Paper Cost" on the TREC-5 collection. The top fifty documents retrieved are presented as the ranked list of titles and fifty spheres corresponding to each document. "Paper Cost" is the TREC topic 286. Figure 6.2 gives a complete description of the topic stating what documents are considered relevant. We see that is a fairly complex search request.

The ranked list is broken into two columns with 25 documents each on the left and on the right side of the screen with the spring-embedding visualization in the middle. The list flows starting from top left corner down and again from the top right corner to the bottom of the window. The pages are ranked by the search engine in the order they are presumed to be relevant to the query. The rank number precedes each title in the list.

The spring-embedding visualization, or the configuration of fifty spheres, is positioned between the two columns of titles. This organization makes the user focus on the visualization as the central part of the system. The spheres appear to be floating in space in front of the ranked list. We believe that such an approach allows us to preserve some precious screen space and at the same time it stresses the integration of the ranked list and the visualization.

Each sphere in the visualization is linked to the corresponding document title in the ranked list so clicking on the sphere will select the title and vice versa. Selecting a document puts a black outline around the corresponding title and sphere – e.g., the document ranked 1 in Figure 6.1. The user can examine the embedding structure and place it in the best viewing angle by rotating, zooming, and sliding the whole structure while dragging the mouse pointer. (Only the spheres can be manipulated in this fashion – the ranked list remains in place.)

If the user points to a document title or a sphere with the mouse pointer while keeping a control key pressed, a small window similar to a comic balloon pops up showing the document description (Figure 6.1). The content of that window is the document title and description paragraph (or snippet) if one is returned by the search engine for the document. INQUERY returns only document

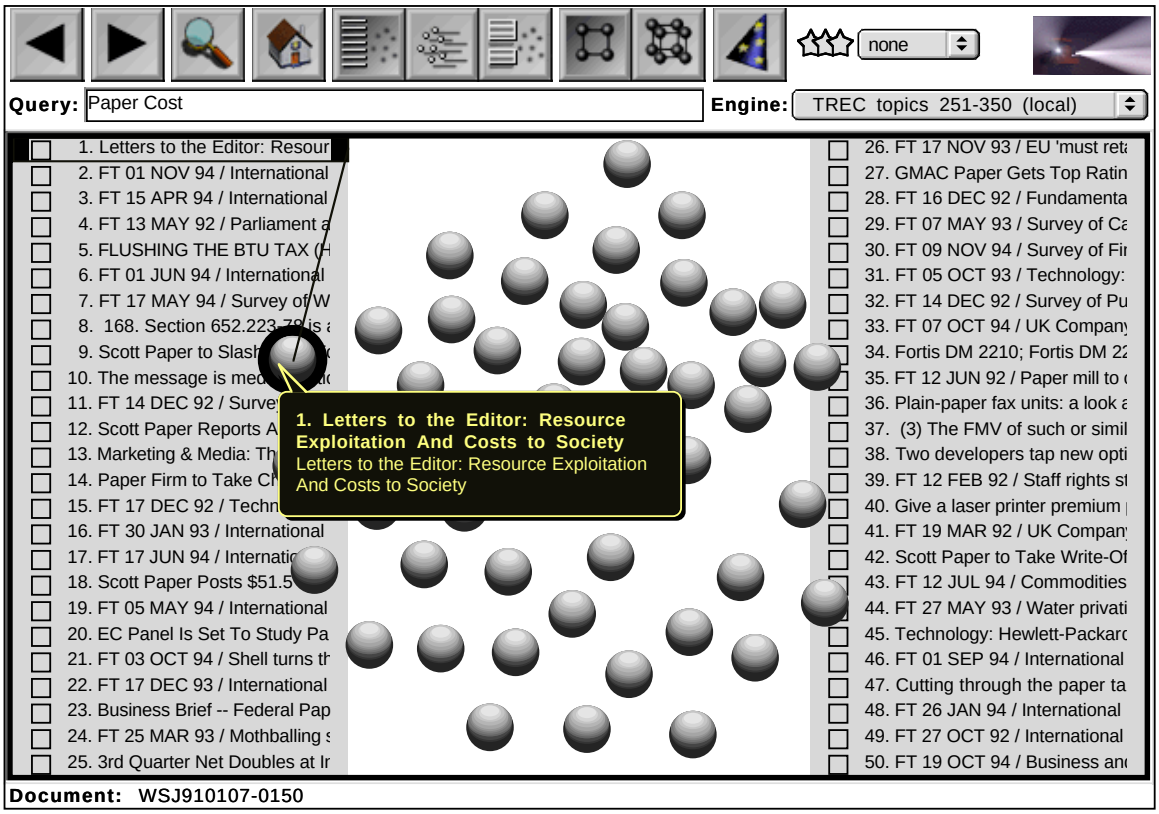

Figure 6.1. Screen shot of the Lighthouse system. The top fifty documents retrieved by the INQUERY search engine for the "Paper Cost" query. A two-dimensional picture is shown.

titles so the snippet in our example is a copy of the title. In addition a line connects the sphere and the title. This design preserves screen space and keeps the snippet readily available to the user by a gesture with a mouse. The line literally links the two document representations – the title and the sphere – together. A double-click on the document title (or sphere) opens the document in the web browser.

The same set of spheres can appear as either a 3-dimensional (Figure 6.3) or 2-dimensional (Figure 6.1) structure. The user can switch the dimensionality on the fly by selecting the button in the toolbar at the top of the window. We achieve the effect of depth in the visualization by using perspective projection of the spheres – the remote spheres appear smaller than their front counterparts – together with the fog effect – the color of the remote spheres is closer to the background color than the color of the front spheres.

Additionally, Lighthouse has an option to present the spheres as floating on top of a plane. Each sphere casts a dark shadow on that plane (Figure 6.3). The intensity of the shadow is proportional to the height of the sphere above the plain: the spheres at the bottom of the picture cast very dark shadows, while light-grayed shadows correspond to the spheres at the top of the structure. The perspective projection of the plane creates additional depth clues in the image. Also a mouse click at a sphere shadow selects both the sphere and the document title.

6.1.1 Placing Titles into Embedding

An alternative to the static arrangement of document titles in the ranked list on the sides of the window is attaching the titles to the document spheres directly (Figure 6.4). This arrangement creates a sophisticated first person fly-through environment immersing the user in the document configuration. It also labels the spheres, allowing the user to determine the corresponding document

Number: 286

Topic: Paper Cost

Description: Why is the cost of paper rising?

Narrative: (defines a relevant document)
In the last year or so the publishing industry is said to have faced a 40% increase in the cost of paper. What factors have led to this price rise? Is there a shortage of materials? Has the cost of processing risen? Have higher taxes been imposed?

Figure 6.2. TREC topic 286, "Paper Cost".

titles for multiple spheres at one quick glance. This title placement eliminates an extra interface action – control key pressed while moving mouse pointer over a document sphere to get the snippet balloon – required to establish the connection between a sphere and a document title.

While such an interface has a compelling advantage, its usability remains an open question. Our experience suggests that adding another fifty objects – document titles – into the visualization tends to clutter the picture. It makes the task of determining the correct proximity relationship between individual spheres more difficult. We believe that an experienced searcher may benefit from having both of these approaches at her fingertips.

6.1.2 Integration of Clustering

There is a third alternative for arrangement of document titles in Lighthouse (Figure 6.5). The titles remain in two columns around the embedding configuration similar to the ranked list arrangement. The individual titles are grouped together following the Ward clustering algorithm and forming the cluster list as described in Chapter 3. The clusters are ordered using the ranks of the highest ranked document in each cluster.

Lighthouse visualizes the cluster list by placing a rectangular bracket or "handle" next to individual cluster (see Figure 6.6 for a zoomed out insert from Figure 6.5). A mouse click on a cluster handle selects all documents in the cluster highlighting their locations in the embedding.

6.2 Interaction Example

A user's interest or the relevance assessment of the document is expressed by clicking on the checkbox attached to each document title. One click marks the document as non-relevant, the corresponding title and sphere are highlighted in red. A second click marks the document as relevant and both the sphere and the title show up in green. Another click removes the mark from the document.[1] Alternatively, a right mouse button click on a document object brings up a menu box where the user can select the relevance mark for the document.

Figure 6.7 shows the document set from Figure 6.1 with the top three documents in the ranked list marked with their relevance judgments. The documents ranked 1 and 2 are non-relevant and colored red. The next document – ranked 3 by INQUERY – is relevant and it is shown in green.

The results from Chapter 4 suggest that the best course of action for a user looking for relevant material is to focus on the embedding and examine the documents in the proximity of the green sphere. The user may elect to handle this task herself. Lighthouse, however, can provide a very effective assistance.

Given the ranking information obtained from the search engine and the relevance judgments collected from the user, Lighthouse estimates the expected relevance values for the unjudged docu-

[1] The selection of colors reflects a common idea in the western world of green as equivalent to "go" and red as a synonym of "stop". The colors can be easily changed to reflect any other scheme using the preference commands.

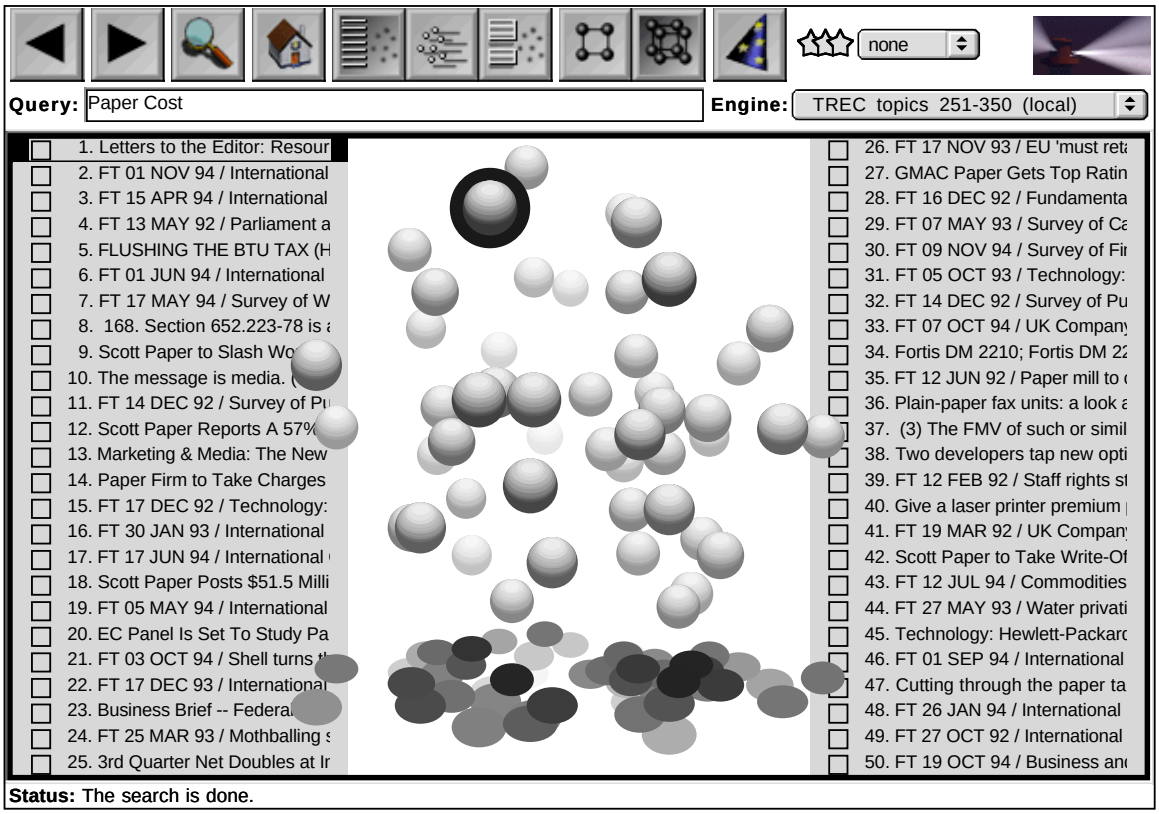

Figure 6.3. Screen shot of the Lighthouse system. The top fifty documents retrieved by the INQUERY search engine for the "Paper Cost" query. A three-dimensional pictures is shown. The document titles ordered by the search engine – the ranked list order.

ments. Currently the system calculates the values of $F_{AC}(\mathcal{D}_t, d)$ for each document (see Chapter 5). We do not use the most effective $F_{Tile}(\mathcal{D}_t, d)$ because the performance differences between two search strategies are small and the latter function is more computationally expensive.

Lighthouse has two different tools to convey the relevance estimates to the user. Both tools operate in suggestion mode – they point the user to supposedly interesting material without forcing their choices on him. Both tools can be switched on and off using the controls in the toolbar at the top of the window.

6.2.1 Shade Wizard

The first tool, the *Shade Wizard*, indicates the estimated relevance for all unjudged documents by means of color and shape. Specifically, if the system estimates the document is relevant, it highlights the corresponding sphere and title using some shade of green. The intensity of the shading is proportional to the strength of the system's belief in its estimate – the more likely the document is relevant, the brighter the color. The same is true for estimated non-relevant documents – the more likely the document is non-relevant, the brighter the red shade of the corresponding object on the screen. The same color shade is used to highlight the document title backgrounds. Additionally, the length of that highlighted background is proportional to the strength of the system's belief in its estimate. The highlighted backgrounds in the left column are aligned on the left side and the highlighted backgrounds in the right column are aligned on the right side. Note that a white sphere and a very short highlighting for the document title reflects that the system's estimate of that document relevance is almost exactly between "relevant" and "non-relevant" – i.e., even odds

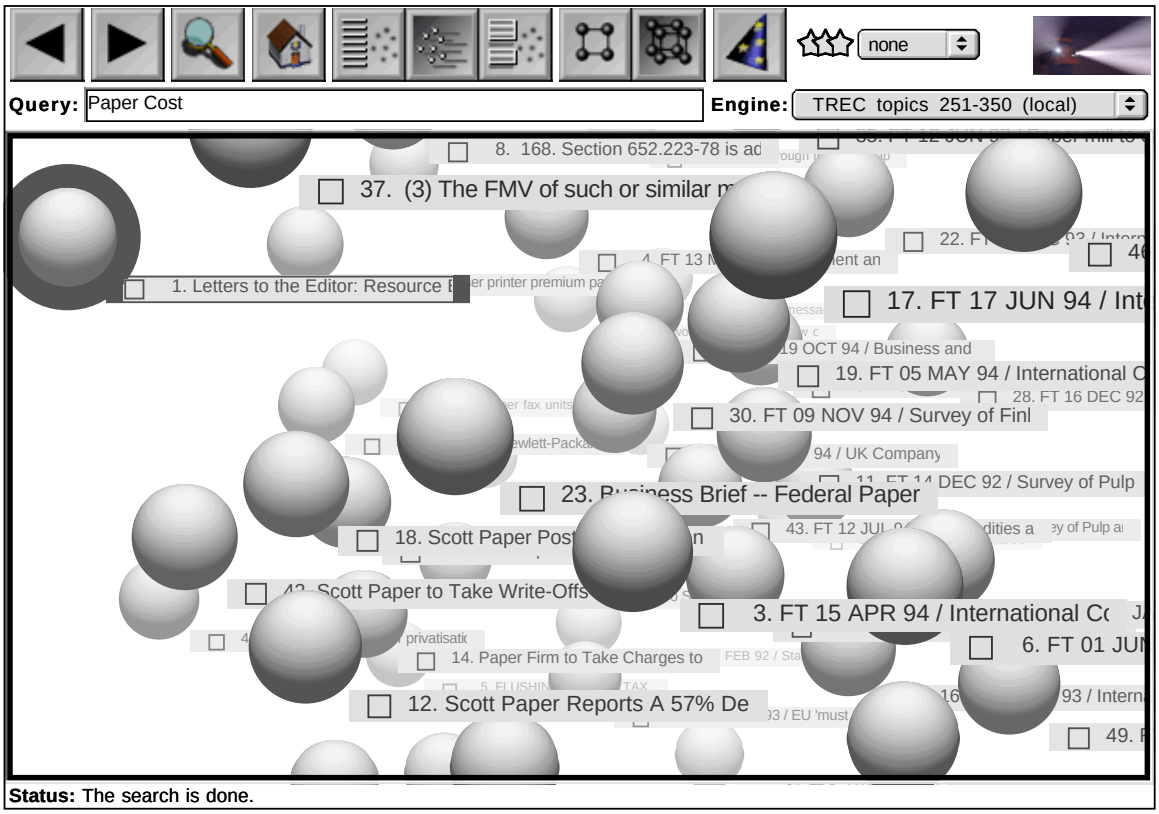

Figure 6.4. Screen shot of the Lighthouse system. The top fifty documents retrieved by the INQUERY search engine for the "Paper Cost" query. The document titles are placed into the spring-embedding visualization for the "fly-through" mode as a three-dimensional picture.

that the document is relevant. The unjudged document titles are further separated from the judged documents by using a gradient fill for painting their background.

Consider the example on the Figure 6.8. We see the same fifty documents retrieved by the "Paper Cost" query with the top three documents judged. The Lighthouse estimate for the rest of the document set is rather "pessimistic" – most of the documents are expected to be non-relevant. There are several white-colored documents (those ranked 29, 32, 33, and 50). There is also one slightly green document ranked 19 in the list. Its title background is shaded for about one third of its length. We highlighted the title and sphere with the black outline. One quick look gives us a pretty good idea where in the embedding the user can expect to find relevant documents – the white spheres are located above the center of the configuration and next to the green sphere.

Note the obvious lack of symmetry among green and white spheres in the picture. One might expect these spheres to form a circle around the judged green sphere. Recall that the $F_{AC}(\mathcal{D}_t, d)$ values are computed using the distances in the high-dimensional document vector space. Visualizing the same documents in two dimensions creates distortions in the picture. We can think about that white area as a projection of a multidimensional spherical cluster onto the two-dimensional visualization space. Additionally, $F_{AC}(\mathcal{D}_t, d)$ takes into account proximity to the known non-relevant documents and the original ranking. The red spheres occupy the top of the picture on both sides. That is why the white spheres seems to be floating down the middle of the structure.

6.2.2 Star Wizard

Figure 6.9 shows the same document configuration after we followed the Shade Wizard's advice and examined the document ranked 19. It is relevant and marked green in the visualization. Light-

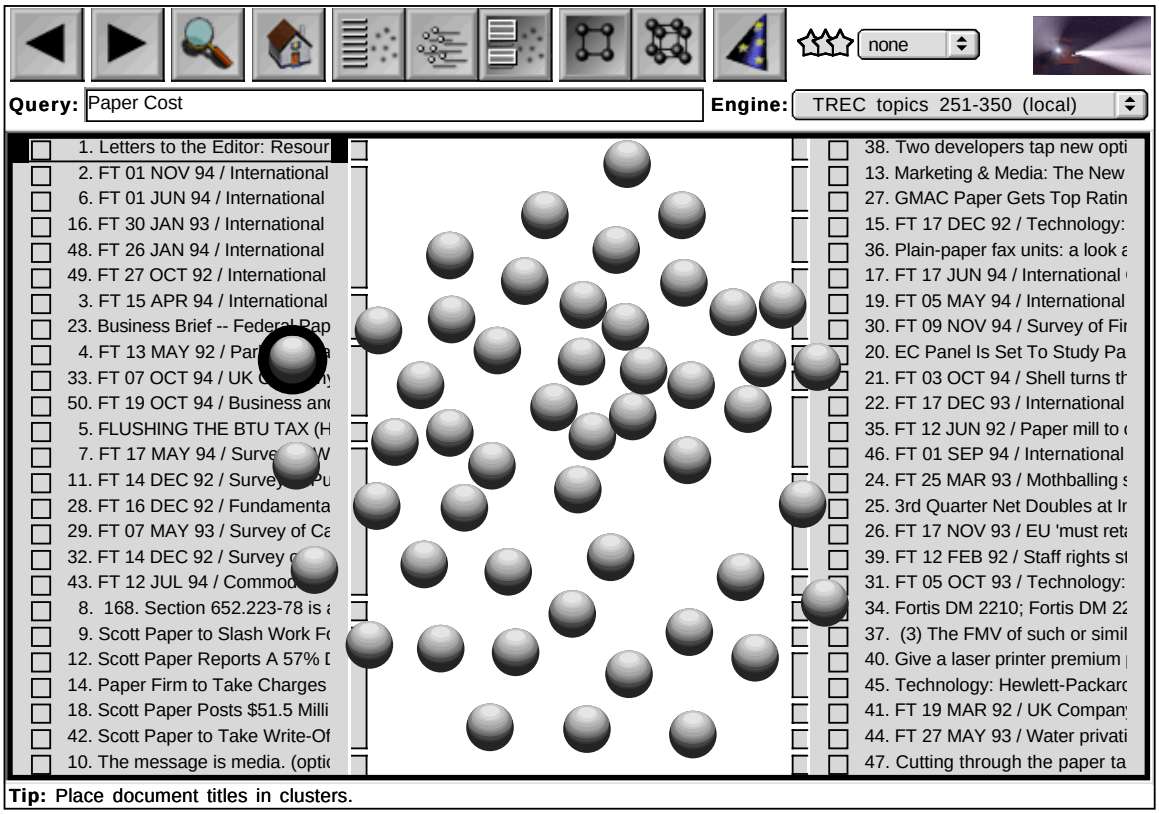

Figure 6.5. Screen shot of the Lighthouse system. The top fifty documents retrieved by the INQUERY search engine for the "Paper Cost" query. A two-dimensional pictures is shown. The document titles are placed in clusters created by the Ward algorithm.

house raised its evaluation of the document set – there are now more spheres estimated as relevant and shaded green or white.

Our experience suggests that it can be very difficult to exactly discriminate between several documents with similar relevance estimates – when the documents are painted with what looks like the same shade of green and even the title backgrounds are of the same length – e.g., documents ranked 11 and 30 on the screen shot. We introduce the second tool that we call the *Star Wizard*. It is controlled by the popup button in the window toolbar. It elaborates on the same information used by the Shade Wizard and indicates the three documents with the highest estimate of relevance. The highest ranked document is marked with three stars (document ranked 30 on the screenshot), the next one with two (ranked 11), and the third one is marked with one star (ranked 33). The stars are placed both by the corresponding document sphere and at the start of document title.

While the Shade Wizard provides a global overview of how the relevance estimates are distributed in the document set, the Star Wizard points the user directly to the most likely relevant documents.

6.2.3 Relevant vs. Non-relevant Separation

We repeatedly examined the documents following the Lighthouse's advice and marked each document with its relevance value. Figure 6.10 shows the document set after we looked at the last relevant document. We have looked at 10 non-relevant documents to find 12 relevant ones. Two out of those 10 were considered at the very beginning of the search. Note that we have a relevant document ranked 50 by the search engine. If we were to follow the ranked list, we would have examined 38 non-relevant documents before finding the last relevant one.

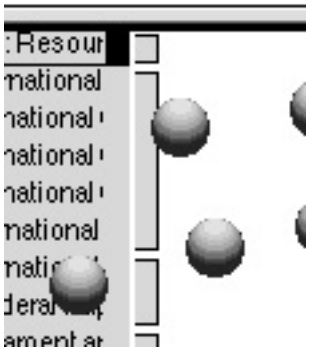

Figure 6.6. Zoomed out insert from Figure 6.5 showing the cluster handle interface object.

This example illustrates a typical situation for the spring-embedding visualization: the relevant documents form a tight group. This group on average is sprinkled with some non-relevant material. Nevertheless, we almost always observe a quite distinct grouping of relevant documents.

This isolation of relevant documents is more apparent in the three-dimensional version of the embedding (Figure 6.11). There is a distinct gap on the shadow plane between two group of shadows – one from the cluster of examined documents and the other from the rest of the spheres.

6.2.4 Clustering

Figure 6.12 shows the same set of documents as Figure 6.10 with the documents arranged into clusters as on Figure 6.5. We see that there are distinct, pure relevant clusters in the list, i.e., the groups containing only relevant documents. There are two such clusters in the set: one starts with the document ranked 4 and the other begins from the document 17. These results are in full agreement with the Cluster Hypothesis, specifically the clustering algorithm brings the relevant documents together. If the user finds a single document from such a cluster, she would be able to examine several of relevant documents without wasting her time on non-relevant material.

The problem is to locate the relevant clusters. Figure 6.12 demonstrates that sometimes a relevant cluster may appear quite far down the list, e.g., the cluster with document 17. Following the strategy outlined in Chapter 3 one would have to examine at least 11 documents – one from every preceding cluster – before finding it.

The clusters containing a mix of relevant and non-relevant material can be observed frequently. There are four such clusters in this example: clusters that start with documents 2, 3, 7, and 22. We selected the cluster that starts with document 2 and the corresponding documents have a thick black outline on Figure 6.12. Document 16 is in that cluster and it is relevant. It seems like this document should not have been placed in that cluster. The spring-embedding visualization reveals that document 16 is similar to both non-relevant (document 48, the closest red sphere with black outline) and relevant (document 17, a green sphere to the left and above it) documents. It also looks like it is more similar to the non-relevant document, but apparently it has enough material resembling the content of document 17 to be judged relevant. This is an example of the clustering algorithm making a hard choice and drawing a border between two classes based solely on the inter-document similarity. Thus, while the clustering allows us to locate groups of similar documents quickly, the spring-embedding shows how the individual documents from different clusters relate to each other.

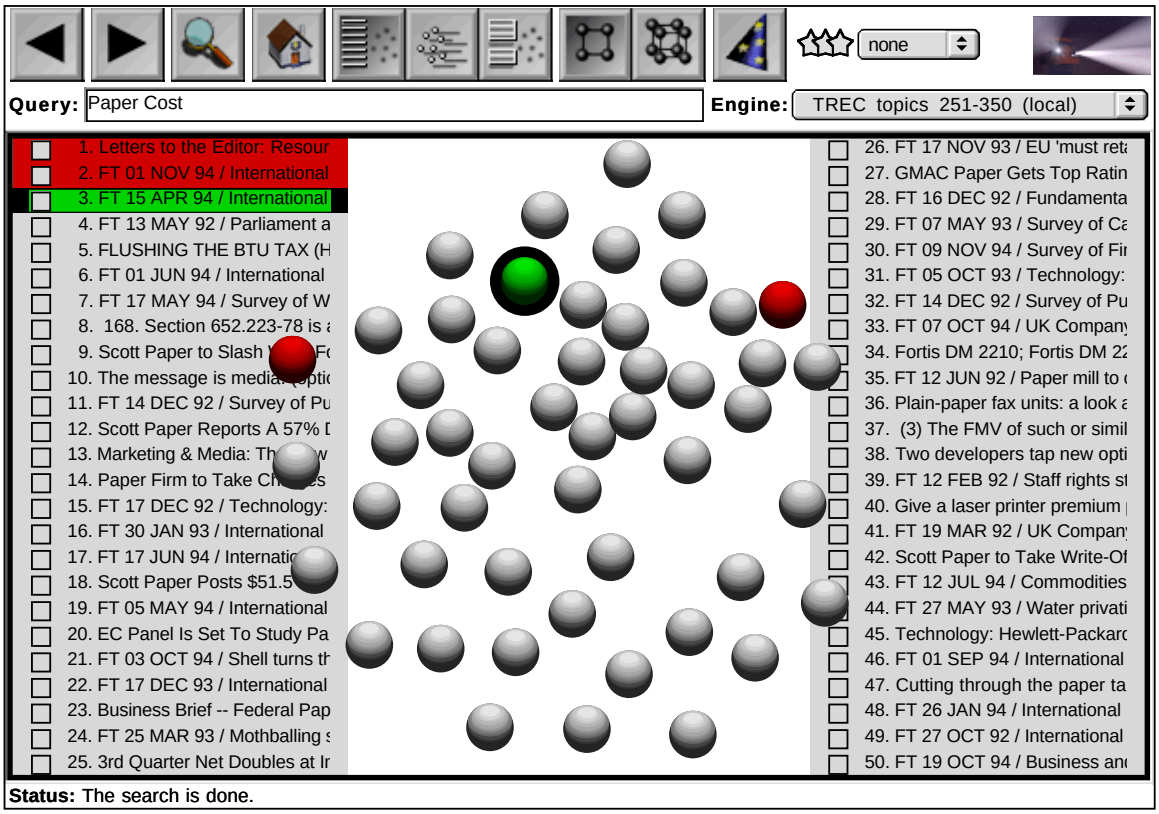

Figure 6.7. Screen shot of the Lighthouse system showing the highest ranked relevant document (green) and all preceding it non-relevant documents (red) for the "Paper Cost" query.

6.2.4.1 Adaptive Clustering

The clustering problems become more apparent when we enable another feature in the Lighthouse clustering algorithm called *adaptive clustering*. Given available relevance judgments, the clustering algorithm attempts to select the threshold that separates relevant from non-relevant documents. The algorithm starts with each document assigned to a unique cluster and proceeds to merge the closest pair of clusters. The algorithm stops if the resulting cluster will contain both judged relevant and non-relevant documents. We have observed that the resulting clustering structure almost always contains only singleton clusters – clusters with a single document. A perfect threshold that clearly separates all relevant from non-relevant documents does not exist.

6.3 Multiple Aspects

In this thesis we have considered only two possible classes for the retrieved documents: the relevant and non-relevant documents. In practice, user's evaluation of the results returned by a search engine may not fit this simple picture. Imagine a user who is interested in recent advances in electric automobiles. There is no single document that will cover all aspects of this topic. There are documents about new, efficient electric engines. There are documents about advances in battery technology. There are documents about government regulation and support for development of electric cars. All these documents are relevant, but all of them are different so there will be no single relevant group in the spring-embedding visualization. How do we help user to locate all interesting information in this case?

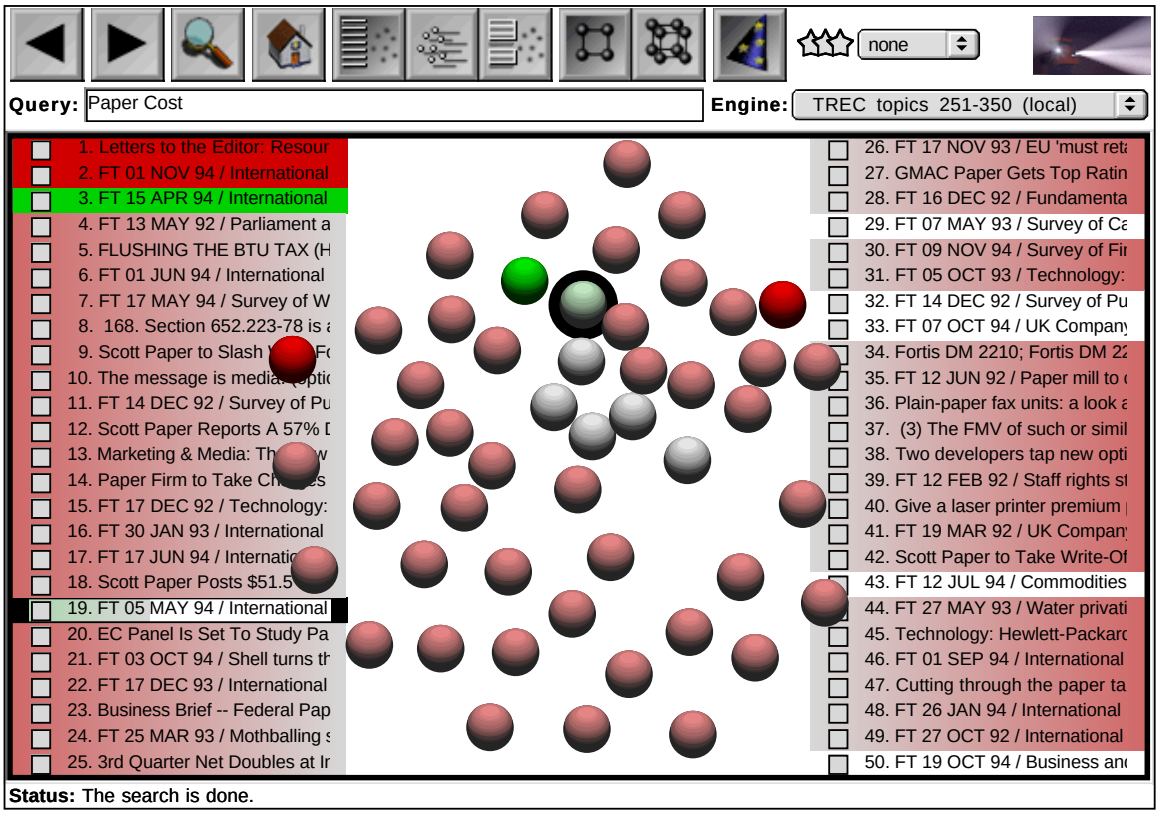

Figure 6.8. Screen shot of the Lighthouse system showing the highest ranked relevant document (green) and all preceding it non-relevant documents (red) for the "Paper Cost" query. The Shade Wizard is switched on.

The problem becomes more complex if the user's search goal is different and she is only interested in one single document for each individual aspect (or subtopic) [58]. She only wishes to see one document about batteries, one about engines, and so on. We have extended Lighthouse to allow the user to assign multiple topics to the documents.

6.3.1 Non-overlapping Aspects

Figure 6.13 shows the results of the query "Samuel Adams" that we ran on the AltaVista search engine. Running the query on such a large and diversified collection as the World Wide Web produces several different topics that all contain words "samuel" and "adams". If we are to consider relevant all the documents that in some way relate to the revolutionary and beermaker Samuel Adams, we will see that there are at least two distinct subtopics: the one about his role in the revolution and another about the Boston Beer Company.

Since Lighthouse allows distinct colors to be assigned to individual subtopics, we marked green several documents that mention the "Samuel Adams" beer brand. The yellow color is assigned to documents about the revolution. There is also a large group of documents that are about other people named Samuel Adams and they have nothing to do with the topic of our search. We marked one of them (document 17) red. Thus there are three distinct classes of documents in this visualization and we are interested in two of them. Star Wizard tool allows the user to locate the most likely candidates for each cluster among the non-examined document. Additionally, we can ask the system to point us to the documents that are most different from those that we have already examined in hope to locate another subtopic in the retrieved set.

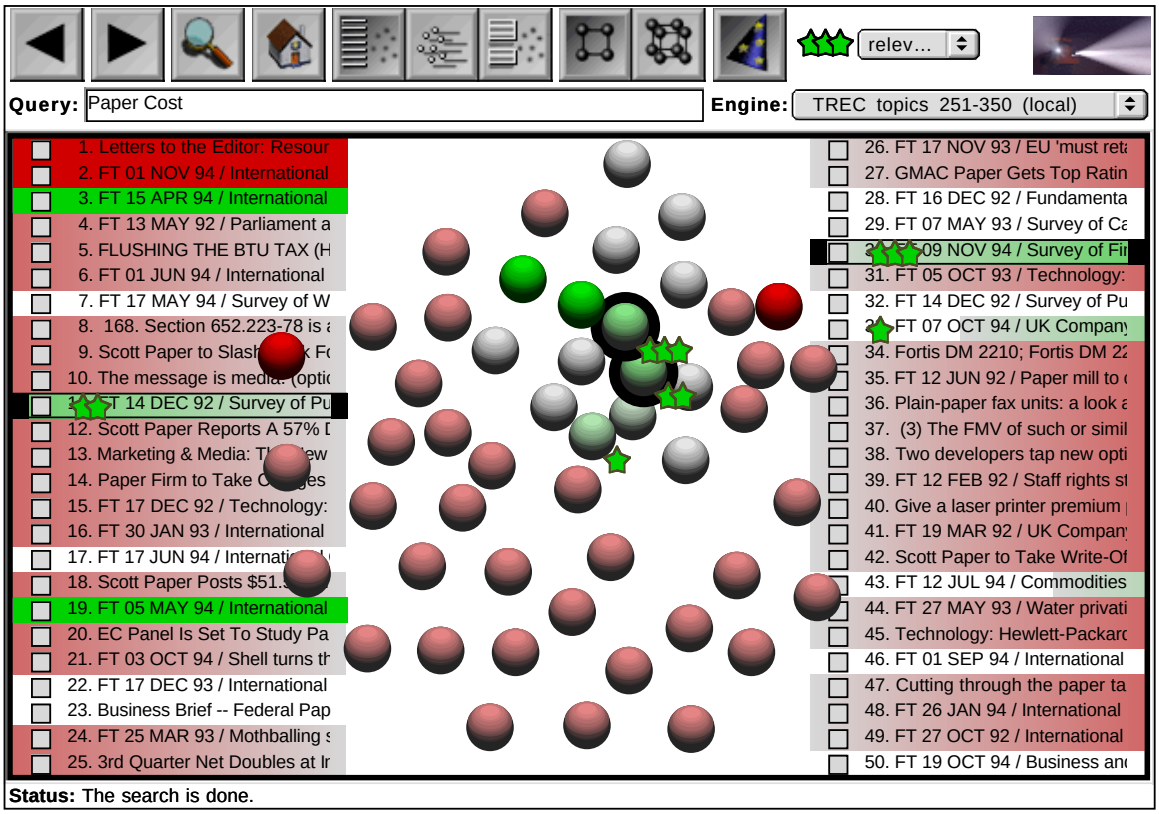

Figure 6.9. Screen shot of the Lighthouse system. Shade Wizard and Star Wizard are switched on.

6.3.2 Overlapping Aspects

The idea of aspects in the preceding section may be expressed as the traditional non-overlapping clustering where each document is assigned to only one cluster or subtopic. Lighthouse is designed to allow for documents that may contain multiple subtopics. Figure 6.14 shows an example of the query "Negative Reactions to Reduced Requirements for College Undergraduate Core Studies" the TREC-6 data set [46]. There are 39 documents in the document set and they discuss seven different aspects of this topic.

As in the previous example a unique color is assigned to each subtopic. However, instead of a solid sphere, each aspect is designated by a pie slice on the sphere surface. The position of the pie slice and its color marks the subtopic label assigned to the document. This information is also indicated with a color stripe in the document title. For example, document 29 discusses two aspects of the request and it is assigned two colors: blue and blue-green. The red color is assigned to documents that does not contain any relevant material.

The documents that discuss the same subtopic group together. For example, there is a group of documents with the blue-green colored slices in the left half of the picture (the blue-green pie slice is in the position corresponding to "7 o'clock" on a clock face). Another group of documents with the green pie slices ("1 o'clock") occupies bottom right quadrant of the picture. There is a third group of five documents that have a yellow-green pie slice ("6 o'clock") at the top of the picture. Notice that documents with two subtopics (blue-green and yellow-green) are placed between the corresponding groups for each of the aspects.

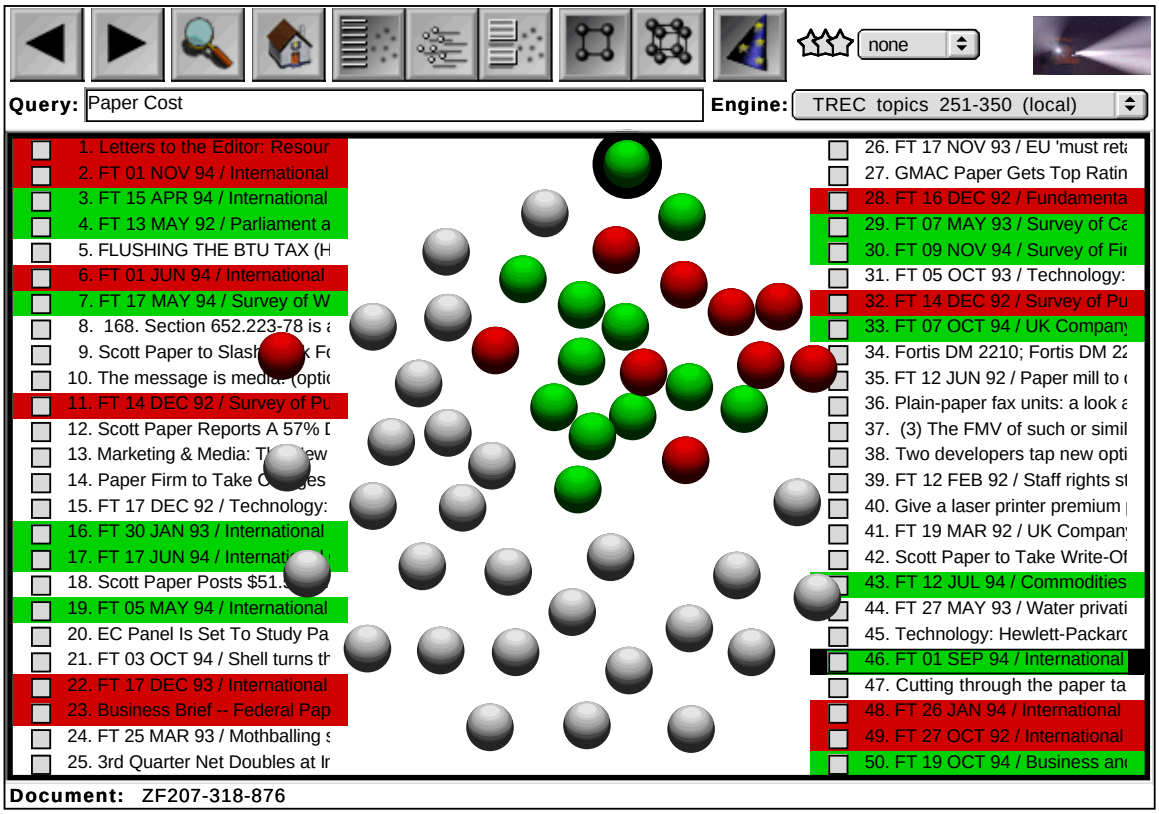

Figure 6.10. Screen shot of the Lighthouse system after all relevant documents were examined following the Star Wizard recommendations. The two-dimensional picture is shown.

6.4 Time-based data

Lighthouse found another application in visualizing clusters of news stories [38]. Figure 6.15 shows a familiar picture where each sphere is representing a group of documents instead of single document. The fifty largest clusters from the collection of news stories provided by Linguistic Data Consortium (LDC) [26] are visualized in two dimensions. This version of Lighthouse is created to help the user track changes and following the progress of news events over time. As new stories arrive from various sources, they are assigned to existing clusters or form new clusters and the picture is adjusted to accommodate changes in the inter-cluster similarities.

The system updates its visualization at discrete time steps. The new data is shown as red slices appearing on the spheres. The amount of the red color is proportional to the percentage of newly arrived documents in that cluster. The largest possible size of a red pie slice on a sphere is exactly one half of the surface. This leaves some space for a user-assigned color label for that cluster. A sphere half covered in red indicates a cluster of stories the user has not seen before the update. There are seven such clusters on that figure. The same novelty information is depicted by partially filling the checkbox at the cluster's title. A completely filled red square indicates a new cluster, e.g., the cluster ranked 24 is a new cluster.

As in Section 6.3.1 we have assigned color labels to several clusters. On this screenshot the green clusters discuss sporting events, the yellow spheres are about financial news, and blue mark news stories about President Clinton and the impeachment process. A user who has assigned a number of colors to various clusters can quickly evaluate new information as soon as it becomes available. For example, in the picture there are two half-red spheres among the green spheres. These spheres correspond to a pair of completely new clusters. Their proximity to the sport clusters indicates that

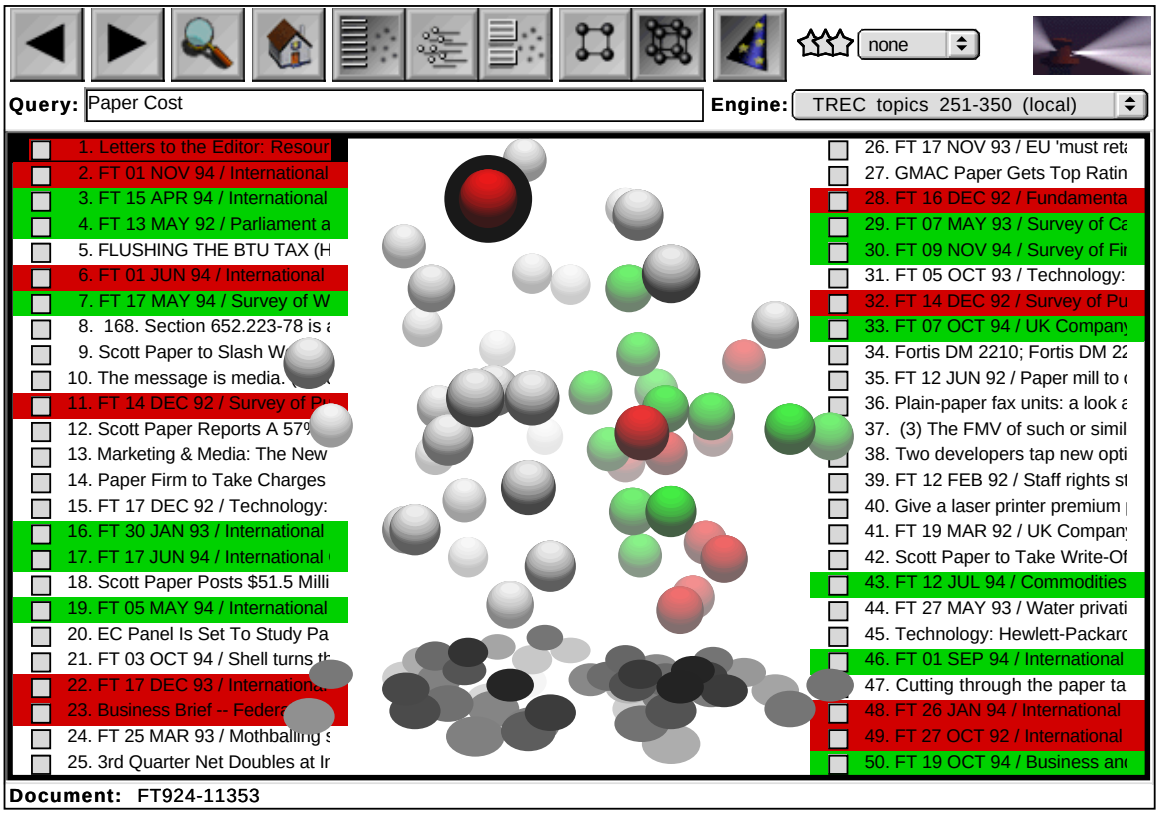

Figure 6.11. Screen shot of the Lighthouse system after all relevant documents were examined following the Star Wizard recommendations. The three-dimensional picture is shown.

these are probably about new sporting events that happened since the last visualization update. Some green spheres have large red pie slices indicating that the corresponding sport topics continue to evolve and expand.

6.5 Implementation

We have implemented the Lighthouse system following the client-server model. The client accepts the query and transmits it to the server. The server forwards the query to the search engine, collects the results as a list of URLs and descriptions in HTML format, parses these results, collects the corresponding web pages, parses and indexes the text of each page, computes the similarities between pages, generates the configurations for both 2- and 3-dimensional visualizations, and returns this data to the client. The server is written in Perl and C. It takes about a second to parse and index the documents, and another 0.5 sec to generate the spatial configuration on a computer with 600MHz Alpha CPU. The total time of a retrieval session is generally between 50 and 100 seconds, where most of the time is spend accessing the search engine and downloading the web pages. The efficiency of course depends on the current network congestion. The client side is written in Java (language version 1.1) and handles all the interaction between the system and the user including the necessary computations for the wizard tools. It can be installed and run locally as an application or it can be downloaded on the fly and run in the browser as an applet. The system is located at our web site [68]. Note that the server processes only one query at a time to avoid overloading the system.

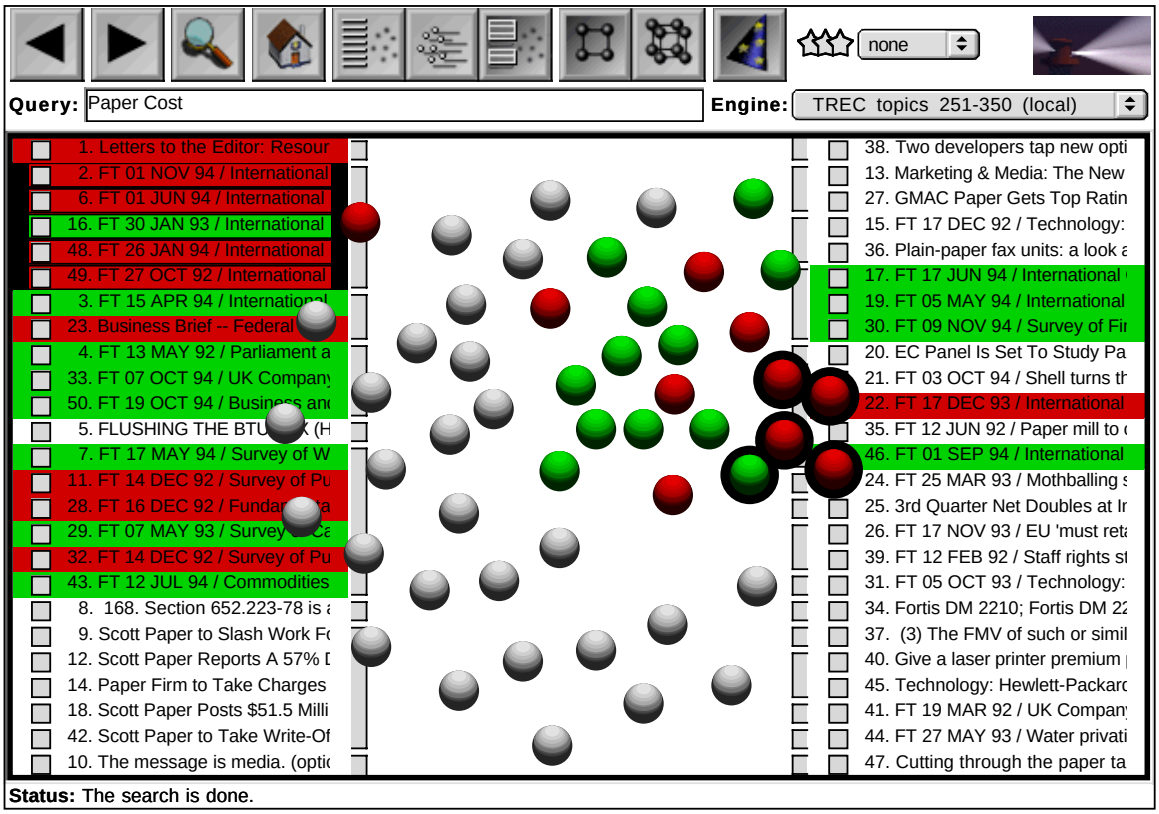

Figure 6.12. Screen shot of the Lighthouse system after all relevant documents were examined following the Star Wizard recommendations. The document titles are arranged in clusters.

6.6 Summary

We have described Lighthouse, an interface system for an on-line search engine that integrates the traditional ranked list, clustering approach and the spring-embedding visualization. Lighthouse displays documents as spheres floating in 2- or 3-dimensional visualization space positioned in proportion to the inter-document similarity. The system accepts user relevance judgments and estimates the relevance values for the remainder of the retrieved set using the search strategy function developed with the TD-learning algorithm. Lighthouse includes two wizard tools that present these relevance estimates to the user using color, shape, and symbolic markings, directing her towards the most likely relevant documents. The system handles multiple topic assignments to individual documents and can visualize and track time-related changes in the document set. Our experience suggests that Lighthouse is fast and can be deployed in the web-based on-line settings.

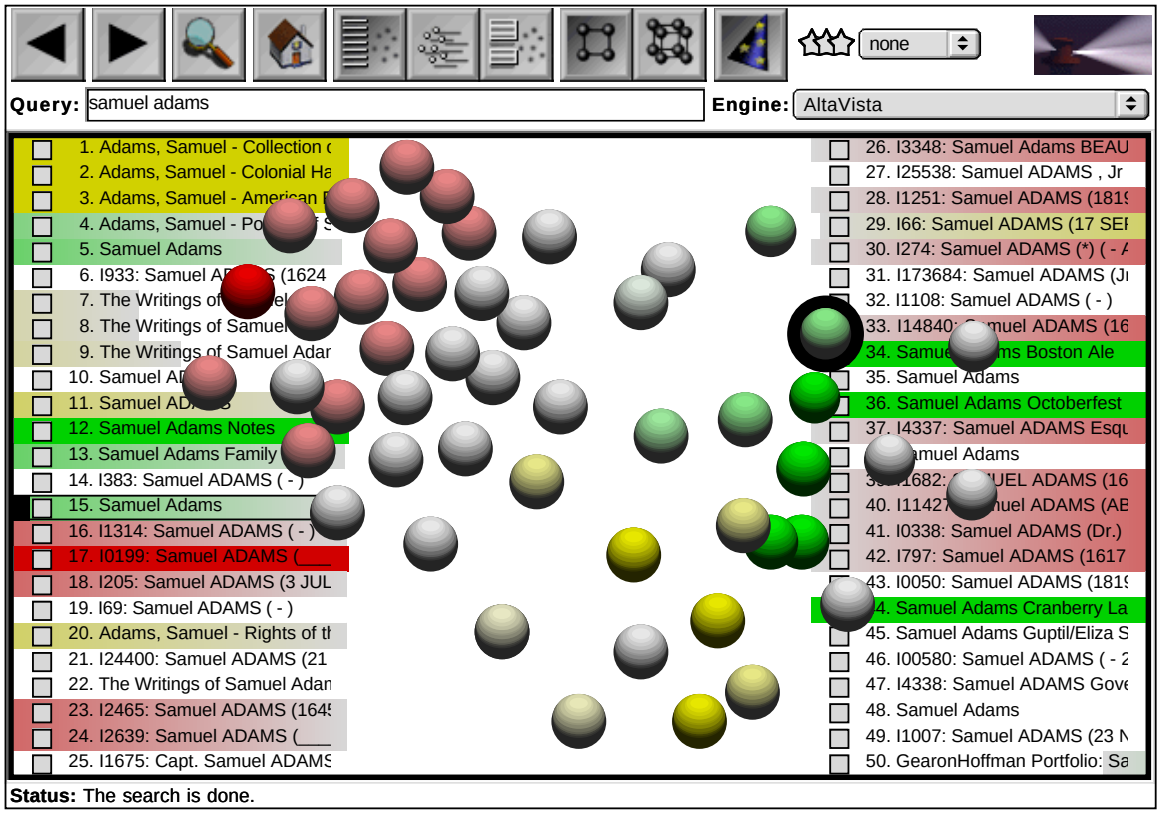

Figure 6.13. Screen shot of the Lighthouse system showing the top fifty documents returned by the AltaVista search engine in response to the "Samuel Adams" query. Three different topics are assigned to the documents.

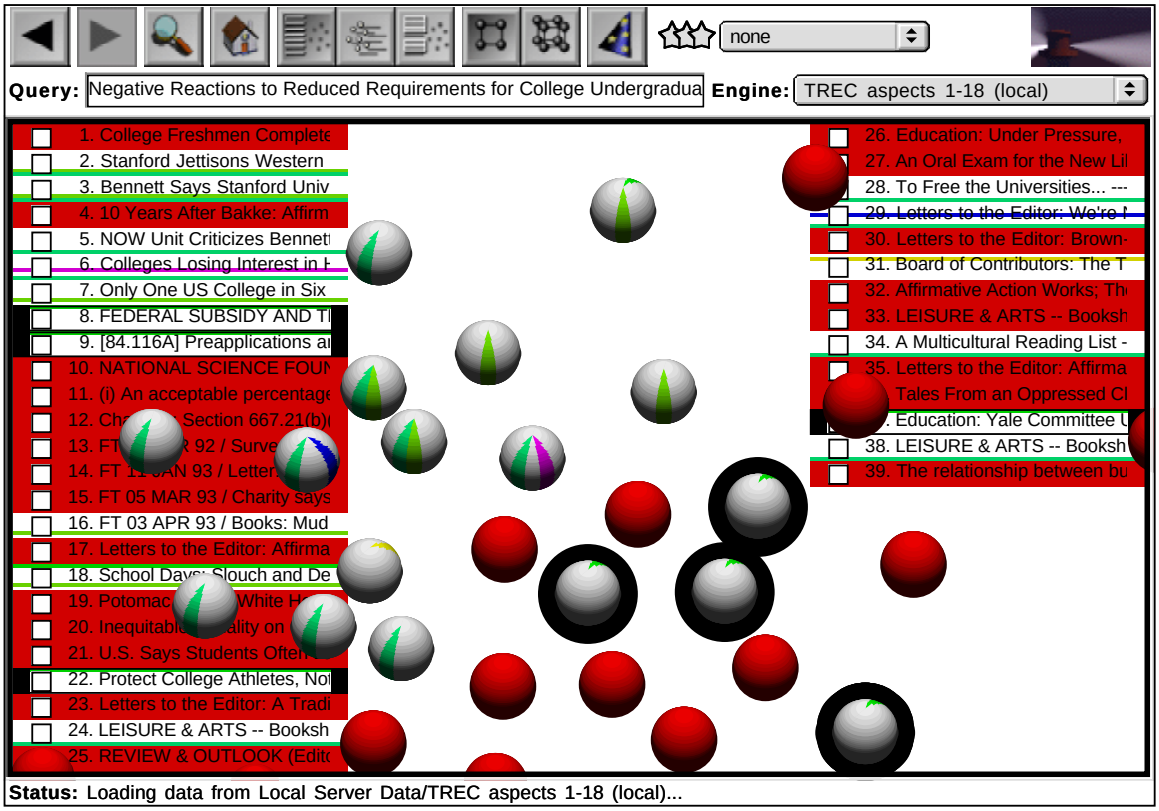

Figure 6.14. Screen shot of the Lighthouse system showing 39 documents from TREC-6 collection in response to the "Negative Reactions to Reduced Requirements for College Undergraduate Core Studies" query. It Illustrates assigning multiple topics to one document.

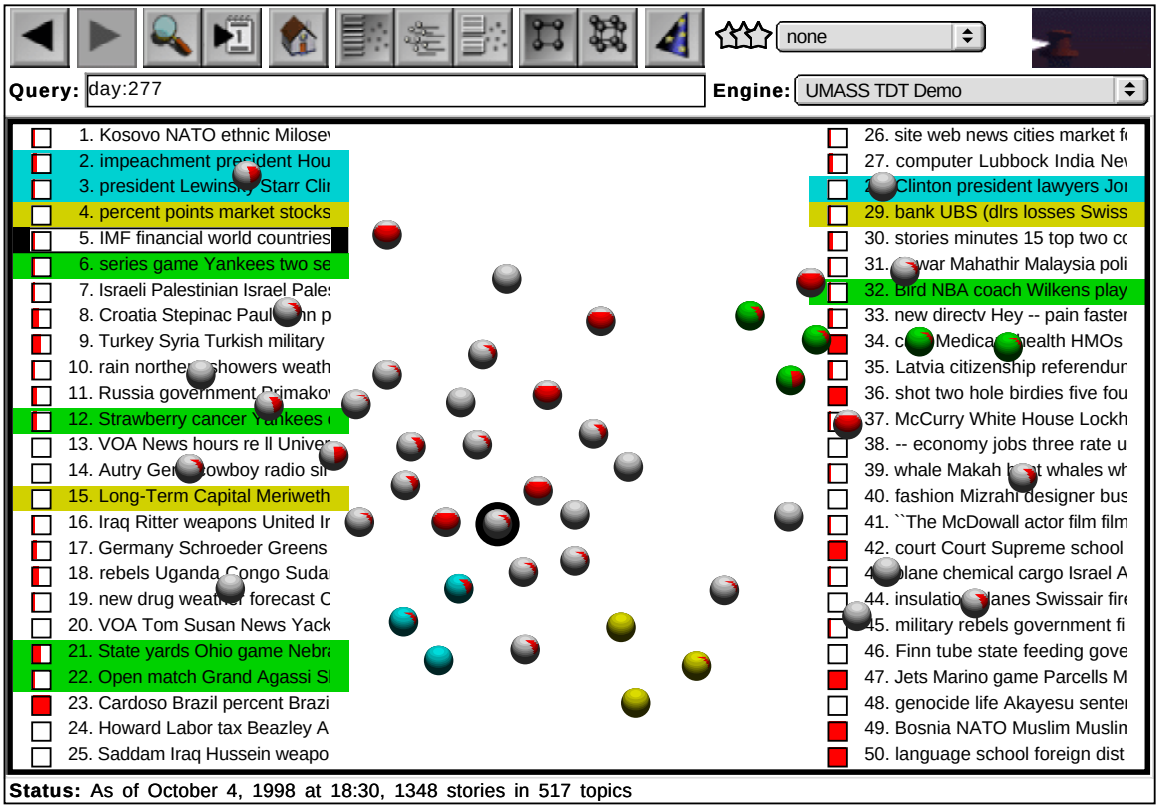

Figure 6.15. Screen shot of the Lighthouse system showing fifty largest clusters from the TDT collection on day 277.

CHAPTER 7

CONCLUSIONS

We have considered the problem of organizing documents returned by an information retrieval system in response to user's query. There are six parts to our analysis:

1. We placed document organization in the context of Information Retrieval and focused our analysis on the specific goal of locating the relevant information among the documents returned by a search engine.

2. To conduct our analysis we introduced the Strategy Based Evaluation framework (Chapter 2). The system's performance is estimated by considering a number of algorithmic strategies for operating the system (search strategies) and evaluating their performance on a standard test set. We apply the framework to two different document organization systems.

3. The first system uses a clustering algorithm to partition the document set into well-defined non-overlapping groups (Chapter 3). We compared six different clustering algorithms. We showed that if a person browsing the retrieved set correctly applies the cluster membership information provided by the system and makes a simple adjustments in direction each time a new document is examined, she can find relevant documents significantly faster than by following the original ranked list. However, the simplicity of the clustering organization comes at a cost of sacrificing many details about the inter-document relationships and forces the system designer to make some hard choices in advance – e.g., selecting the clustering threshold parameter.

4. The second system, described in Chapter 4, addresses those problems with the clustering approach. It is based on the spring-embedding visualization algorithm and presents documents as spheres in space, positioning them in proportion to the inter-document similarities. There is no threshold to select and all the individual pairwise inter-document similarities are depicted in the visualization. We showed that there are some discrepancies in the presentation of inter-document similarity information due to additional constraints imposed by the projection of the multidimensional document vectors on 2 or 3 dimensions. These inaccuracies are small and insignificant for the purpose of using the visualization to locate relevant information. However, we observed that untrained users have problems using the system to its full potential.

5. We designed a "wizard" tool to address these problems with the spring-embedding approach and help users to navigate the visualization more effectively (Chapter 5). We defined the problem of browsing the retrieved document set in terms of reinforcement learning and showed that we can develop a very effective search strategy after a modest amount of training. Using just the information about inter-document similarities, this strategy outperforms the traditional relevance feedback approach based on query expansion and deep analysis of statistical features of individual terms.

6. We designed and implemented Lighthouse – a document organization system that illustrates the application of the thesis results in a real-world search system. Lighthouse integrates the ranked list, clustering, spring-embedding, and the feedback wizard trained with the reinforcement learning algorithm. We gave some examples of Lighthouse helping a user solve various search problems.

7.1 Contributions

This thesis made the following broad contributions to the field of information retrieval:

- We studied two information organization techniques: (1) document clustering and (2) spring-embedding visualization. These approaches to document organization are well-known, however, we were the first to study them in the context of ranked document retrieval task. We applied information organization to a set of documents found by an information retrieval system in response to a predefined query. We investigated how the organization helps the user to isolate interesting material in the retrieved set quickly. Also the extent of our analysis and the questions considered are the prominent features of this work.

- We investigated a method for evaluating interactive information organization systems that can be carried out in the absence of a user. We studied applications of this method to measuring the retrieval effectiveness of information organization systems – how well the systems support a user in finding interesting information. This approach is significantly different from what has been attempted before.

Specifically, we contributed the following:

- We defined an interaction model between an information organization system and a user. We designed a novel method for evaluating interactive information organization that does not require immediate user participation – we simulated the user with an artificially constructed (though plausible) user strategy.

- We compared six different hierarchical agglomerative clustering algorithms. We defined an effective method for transforming a clustering hierarchy into a partition of the document set. We showed that these document set partitions can be much more helpful in locating the relevant information than the traditional ranked list.

- We studied a visualization system that places documents in 2- and 3-dimensional space. We were the first to investigate how successful the visualization is in helping a user to locate interesting information in the retrieved document set. We showed that such a visualization can be more beneficial than a traditional ranked list approach for presenting the retrieval results.

- We showed that the spring-embedding visualization of inter-document similarities is accurate enough for the retrieval task. Using the proximity information between objects in the 2- and 3-dimensional projections of the document set to isolate relevant documents was as effective as using the actual inter-document similarities for the same task.

- We showed that while a 3-dimensional visualization was more accurate than a 2-dimensional one, the users were more effective and comfortable with the 2-dimensional interface.

- We stated the task of locating the relevant information in terms of reinforcement learning. We defined an automatic agent that performed this task on a retrieved document set and showed that the browsing effectiveness of such an agent was significantly improved after a modest amount of training.

- We presented a system that integrated the traditional ranked list with document clustering and spring-embedding visualization of the retrieved documents. The system supported the user's browsing process with a wizard tool constructed using the reinforcement learning approach. The system's design was built on the results from the work in this thesis.

7.2 Future Work

This thesis is about information access. The existing tools available on the Internet for a casual user are inadequate for anything but very simple search tasks. If the user's request cannot be satisfied with one document or if this "perfect" document does not appear among the first ten retrieved, a web-based search engine cannot help the user with her search problem. We have considered only one search task in this thesis – locating all relevant information among the top portion of the retrieved document set. Our choice was motivated by the available test data. We believe that many other information access tasks will benefit from properly designed information organization approaches.

In Section 6.3.2 we described how Lighthouse can be used to assist the user in isolating different aspects of the relevant topic. In our informal experiments we observed that documents about different subtopics group together in the spring-embedding visualization and documents containing multiple subtopics are found exactly where our intuition predicted – between the clusters corresponding to individual subtopics. We are looking into adapting the wizard training algorithm for this task and conducting a more formal evaluation.

The information organization approaches considered in this thesis do not mandate a single document as a unit of information. We can use the same techniques to accommodate clusters of documents. For example, in Section 6.4 we applied the system to visualize clusters of news stories. This brings up another question: Can we extend the visualization with the fourth (in 3D) or the third (in 2D) axis representing time? We are looking to include this extra information – inter-cluster similarity in time – and see how it will affect the visualization, wizard tool, and our evaluation.

We are also interested in taking these approaches in the other direction – organizing and visualizing individual paragraphs. Documents that discuss multiple topics create a difficult problem for most of the clustering approaches: it is a hard decision to select the correct cluster for such a document. It seems like the spring-embedding visualization should address this problem by placing the "multi-topic" documents clearly between the "single-topic" documents. However, the dimensionality constraints on the visualization space may easily prevent such an intuitive layout. Breaking the documents into paragraphs and probably clustering the paragraphs to isolate individual topics as units of information is a better solution. We imagine a visual metaphor where a document is represented by a steel rod and the sections of the rod represent the individual paragraphs. The springs connect individual sections and the rods can twist or bend depending on the similarity between consequent sections. Then a multi-topic document will be shown as a rod with sharp bends stretched among several groups.

We are also interested in truly integrating the clustering and spring-embedding approaches. In this thesis we developed effective search strategies for each technique. What if we combine the output of these strategies? We can do that by adding cluster membership as another input feature to the feedback wizard or we can use a boosting algorithm that combines inputs of multiple classifiers [37]. We expect such a combination to perform better than the individual components.

The information organization systems in this thesis use the inter-document similarity information. In contrast to the traditional relevance feedback approach the user's input affects only the direction of the search – the documents representations and inter-document similarities remain constant. It is interesting to consider adjusting term weights and therefore their importance based on the relevance information from the user. This process will modify document representations and change inter-document similarities hopefully pushing relevant documents together and away from non-relevant. This is similar to the relevance feedback algorithm modifying the original query to create a new, better version of the request.

Our user study of the spring-embedding visualization demonstrated that people have significant problems navigating three-dimensional structures visualized on a flat screen. We believe this result requires a further study with better tools for three-dimensional presentation and manipulation. For example, using a virtual reality cave system or a simple pair of stereo-goggles may affect the overall performance.

This thesis is also about evaluating information organization systems. The Strategy Based Evaluation framework is a very quick and cost-effective method for obtaining a low-bound estimate on the system performance. We are looking into extending this analysis to incorporate the textual

information that is normally available to a user in the form of document titles and cluster summaries. For example, we can consider two additional parameters for the search process: one is the probability that a user will correctly recognize that a document is relevant by looking at its title (α). The other is the probability that a user will do the same with a non-relevant document (β). Then we modify our experiments so that a search strategy may rearrange the ranked list it creates: it recursively moves the top ranked document to the end of the list with the probability $1 - \alpha$ if it is relevant and β if it is not. This should simulate a user skimming over a list of document titles and ignoring those she considers non-relevant. What we need is a short user study that establishes the values for α and β. These parameters will vary significantly depending on the collection and summarization method. There have been some studies that measure the quality of different document summarization techniques [98]. We believe that if one system is more effective than the other without the textual information from titles or summaries, it is still going to be more effective if we add the text to both systems.

The SBE framework allows a researcher to isolate and individually study both the system's and the user's effect on the overall performance. We are interested in applying this framework to other systems. We are also interested in comparing this evaluation approach with an extended user study and in comparing the performance predicted by SBE with the performance observed from real users.

BIBLIOGRAPHY

[1] Aalbersberg, IJsbrand Jan. Incremental relevance feedback. In *Proceedings of ACM SIGIR* (1992), pp. 11–22.

[2] ACM Digital Library. `http://www.acm.org/dl/`.

[3] Allan, James. *Automatic Hypertext Construction*. PhD thesis, Cornell University, January 1995.

[4] Allan, James. Incremental relevance feedback for information filtering. In *Proceedings of ACM SIGIR* (1996), pp. 270–278.

[5] Allan, James. Building hypertext using information retrieval. *Information Processing and Management 33*, 2 (1997), 145–159.

[6] Allan, James, Callan, Jamie, Croft, Bruce, Ballesteros, Lisa, Broglio, John, Xu, Jinxi, and Shu, Hongmin. Inquery at TREC-5. In *Fifth Text REtrieval Conference (TREC-5)* (1997), pp. 119–132.

[7] Allan, James, Callan, Jamie, Croft, W. Bruce, Ballesteros, Lisa, Byrd, Donald, Swan, Russell, and Xu, Jinxi. Inquery does battle with TREC-6. In *Sixth Text REtrieval Conference (TREC-6)* (1998), pp. 169–206.

[8] Allen, Bryce L. Cognitive research in information science. *Annual Review of Information Science and Technology 26* (1991), 3–37.

[9] Allen, Robert B., Obry, Pascal, and Littman, Michael. An interface for navigating clustered document sets returned by queries. In *Proceedings of ACM COOCS: Conference on Organizational Computing Systems* (Nov. 1993), pp. 166–171.

[10] Ambrosini, Leonardo, Cirillo, Vincenzo, and Micarelli, Alessandro. A hybrid architecture for user-adapted information filtering on the world wide web. In *Proceedings of the Sixth International Conference on User Modeling* (1997), pp. 59–61.

[11] Balabanovic, Marko. An adaptive web page recommendation service. In *Proceedings of the First International Conference on Autonomous Agents* (1997), pp. 378–385.

[12] Bates, Marcia J. The design of browsing and berrypicking techniques for the online search interface. *Online Review 13* (Spetember 1989), 407–424.

[13] Belew, Richard K. Adaptive information retrieval: using a connectionist representation to retrieve and learn about documents. In *Proceedings of ACM SIGIR* (1989), pp. 11–20.

[14] Belew, Rick. Rave reviews: Acquiring relevance assessments from multiple users. In *Working Notes of the AAAI Spring Symposium on Machine Learning in Information Access*, Marti A. Hearst and Haym Hirsh, Eds. Mar. 1996.

[15] Belkin, N. J. Interaction with text: Information retrieval as information seeking behavior. *Information Retrieval 10* (1993), 55–66.

[16] Berliner, H. On the construction of evaluation functions for large domains. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence* (1979).

[17] Biron, Paul V., and Kraft, Donald H. New methods for relevance feedback: improving information retrieval performance. In *ACM symposium on Applied computing* (1995), pp. 482–487.

[18] Bookstein, Abraham. Information retrieval: A sequential learning process. *Journal of the American Society for Information Science 34*, 5 (1983), 331–342.

[19] Borg, I., and Lingoes, J. *Multidimensional similarity structure analysis*. Springer-Verlag, 1987.

[20] Buckley, Chris, and Salton, Gerard. Optimization of relevance feedback weights. In *Proceedings of ACM SIGIR* (1995), pp. 351–357.

[21] Byrd, Donald, and Podorozhny, Rodion. Adding boolean-quality control to best-match searching via an improved user interface. Tech. Rep. IR-210, Department of Computer Science, University of Massachusetts, Amherst, 2000.

[22] Callan, J. P. Learning while filtering documents. In *Proceedings of ACM SIGIR* (1998), pp. 224–231.

[23] Card, S., and Moran, T. User technology: from pointing to pondering. In *Readings in Human-Computer Interaction: towards the year 2000*, Baecker, Grudin, and Buxton an Greenberg, Eds. Morgan Kaufmann, 1995.

[24] Chalmers, Matthew, and Chitson, Paul. Bead: Explorations in information visualization. In *Proceedings of ACM SIGIR* (June 1992), pp. 330–337.

[25] Choi, Yong S., and Yoo, Suk I. Multi-agent learning approach to www information retrieval using neural network. In *Proceedings of International Conference on Intelligent User Interfaces* (1999), pp. 23–30.

[26] Cieri, Christopher, Graff, David, Martey, Nii, and Strassel, Stephanie. Large multilingual broadcast news corpora for cooperative research in topic detection and tracking: The TDT2 and TDT3 corpus efforts. In *Proceedings of the Second International Language Resources and Evaluation Conference* (2000).

[27] CMAC. `http://www.ece.unh.edu/robots/cmac.htm`.

[28] Conati, Cristina, Gertner, Abigail S., VanLehn, Kurt, and Druzdzel, Marek J. On-line student modeling for coached problem solving using bayesian networks. In *Proceedings of the Sixth International Conference on User Modeling* (1997), pp. 231–242.

[29] Croft, W. Bruce. *Organising and Searching Large Files of Documents*. PhD thesis, University of Cambridge, October 1978.

[30] Croft, W. Bruce, and Thompson, R. H. I^3R: A new approach to the design of document retrieval systems. *Journal of the American Society for Information Science 38* (1987), 389–404.

[31] Cutting, Douglass R., Karger, David R., and Pedersen, Jan O. Constant interaction-time Scatter/Gather browsing of very large document collections. In *Proceedings of ACM SIGIR* (1993), pp. 126–134.

[32] Cutting, Douglass R., Pedersen, Jan O., Karger, David R., and Tukey, John W. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of ACM SIGIR* (1992), pp. 318–329.

[33] Dataware search engine. `http://www.dataware.com/find/default.html/`.

[34] Dubin, David. Document analysis for visualization. In *Proceedings of ACM SIGIR* (July 1995), pp. 199–204.

[35] Dunlop, Mark D. Time, relevance and interaction modeling for information retrieval. In *Proceedings of ACM SIGIR* (1997), pp. 206–213.

[36] Efthimidiadis, E. N. Query expansion. *Annual Review of Information Science and Technology 31* (1996), 121–187.

[37] Freund, Yoav. Boosting a weak learning algorithm by majority. In *Proceedings of the Workshop on Computational Learning Theory* (1990), pp. 202–216.

[38] Frey, David, Gupta, Rahul, Khandelwal, Vikas, Lavrenko, Victor, Leuski, Anton, and Allan, James. Monitoring the news: a TDT demonstarion system. In *Proceedings of the first International HLT Conference* (2001).

[39] Fruchterman, Thomas M. J., and Reingold, Edward M. Graph drawing by force-directed placement. *Software–Practice and Experience 21*, 11 (Nov. 1991), 1129–1164.

[40] Giangrandi, Paolo, and Tasso, Carlo. Managing temporal knowledge in student modeling. In *Proceedings of the Sixth International Conference on User Modeling* (1997), pp. 415–426.

[41] Godlin, R., Gecsei, J., and Pichet, C. Design of a browsing interface for information retrieval. In *Proceedings of ACM SIGIR* (1989), pp. 32–39.

[42] Google. `http://www.google.com/`.

[43] Gori, Marco, Maggini, Marco, and Martinelli, Enrico. Web-browser access through voice input and page interest prediction. In *Proceedings of the Sixth International Conference on User Modeling* (1997), pp. 17–19.

[44] Harman, Donna. Towards interactive query expansion. In *Proceedings of ACM SIGIR* (1988), pp. 321–331.

[45] Harman, Donna, and Voorhees, Ellen, Eds. *The Fifth Text REtrieval Conference (TREC-5)* (1997), NIST.

[46] Harman, Donna, and Voorhees, Ellen, Eds. *The Sixth Text REtrieval Conference (TREC-6)* (1998), NIST.

[47] Harter, S. P. Psychological relevance and information science. *Journal of the American Society for Information Science 43* (1992), 602–615.

[48] Hearst, Marti A. Improving full-text precision using simple query constraints. In *Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval* (1996).

[49] Hearst, Marti A. User interfaces and visualization. In *Modern Information Retrieval*, Ricardo Baeza-Yates and Berthier Ribeiro-Neto, Eds. Addison-Wesley, 1999, ch. 10, pp. 257–324.

[50] Hearst, Marti A., and Pedersen, Jan O. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In *Proceedings of ACM SIGIR* (Aug. 1996), pp. 76–84.

[51] Hemmje, M., Kunkel, C., and Willet, A. LyberWorld - a visualization user interface supporting fulltext retrieval. In *Proceedings of ACM SIGIR* (July 1994), pp. 254–259.

[52] Hendley, R. J., Drew, N. S., Wood, A. M., and Beale, R. Narcissus: Visualising information. In *Proceedings of IEEE Information Visualization* (1995), pp. 90–96.

[53] Hull, David A., Pedersen, Jan O., and Schütze, Hinrich. Method combination for document filtering. In *Proceedings of ACM SIGIR* (1996), pp. 279–287.

[54] Infoseek. `http://www.infoseek.com/`.

[55] Joachims, T., Freitag, D., and Mitchell, T. Webwatcher: A tour guide for the world wide web. In *Proceedings of IJCAI* (1997), pp. 770–778.

[56] Koenemann, Jurgen, and Belkin, Nicholas J. A case for interaction: A study of interactive information retrieval behavior and effectivness. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems* (1996), pp. 205–212.

[57] Kwok, K. L. A network approach to probabilistic information retrieval. *ACM Transactions on Information Systems 13*, 3 (1995), 324–353.

[58] Lagergren, Eric, and Over, Paul. Comparing interactive information retrieval systems across sites: the TREC-6 interactive track matrix experiment. In *Proceedings of ACM SIGIR* (1998), pp. 164–172.

[59] Lance, G. N., and Williams, W. T. A general theory of classificatory sorting strategies: 1. hierarchical systems. *Computer Journal 9* (1967), 373–380.

[60] Leuski, Anton. Learning of position evaluation in the game of othello. Tech. Rep. TR 95-23, Department of Computer Science, University of Massachusetts, Amherst, 1995.

[61] Leuski, Anton. Combining ranked list and clustering: the best of both worlds. Tech. Rep. IR-172, Department of Computer Science, University of Massachusetts, Amherst, 1999.

[62] Leuski, Anton, and Allan, James. Interactive cluster visualization for information retrieval. In *Proceedings of ECDL'98* (September 1998), pp. 535–554.

[63] Leuski, Anton, and Allan, James. Evaluating a visual navigation system for a digital library. *International Journal on Digital Libraries 3*, 2 (2000), 170–184.

[64] Leuski, Anton, and Croft, W. Bruce. An evaluation of techniques for clustering search results. Tech. Rep. IR-76, Department of Computer Science, University of Massachusetts, Amherst, 1996.

[65] Leuski, Anton, and Utgoff, Paul E. What a neural network can learn about othello. Tech. Rep. TR 96-10, Department of Computer Science, University of Massachusetts, Amherst, 1996.

[66] Lewis, David D. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of ACM SIGIR* (1992), pp. 37–50.

[67] Lewis, David D., Schapire, Robert E., Callan, James P., and Papka, Ron. Training algorithms for linear text classifiers. In *Proceedings of ACM SIGIR* (1996), pp. 298–306.

[68] Lighthouse. `http://toowoomba.cs.umass.edu/~leouski/lighthouse/`.

[69] Lin, Xia, Soergel, Dagobert, and Marchionini, Gary. A self-organizing semantic map for information retrieval. In *Proceedings of ACM SIGIR* (1991), pp. 262–269.

[70] Luhn, H. P. Selective dissemenation of new scientific information with the aid of electronic processing equipment. *American Documentation 12* (1961), 131–138.

[71] Maarek, Y. S., and Wecker, A. J. The librarian's assistant: Automatically assembling books into dynamic bookshelves. In *Proceedings of RIAO'94; Intelligent Multimedia Information Retrieval Systems and Management* (1994).

[72] Menczer, Filippo, and Belew, Richard K. Adaptive information agents in distributed textual environments. In *Proceedings of the second international conference on Autonomous agents* (1998), pp. 157–164.

[73] Mirkin, Boris. *Mathematical Classification and Clustering*. Kluwer, 1996.

[74] Ng, Hwee Tou, Goh, Wei Boon, and Low, Kok Leong. Feature selection, perception learning, and a usability case study for text categorization. In *Proceedings of ACM SIGIR* (1997), pp. 67 – 73.

[75] Northern light. http://www.northernlight.com/.

[76] O'Day, Vicki L., and Jeffries, Robin. Orienteering in an information landscape: how information seekers get from here. In *Proceedings of the INTERCHI'93* (Apr. 1993), pp. 438–445.

[77] Pedersen, G. S. A browser for bibliographic information retrieval, based on an application of lattice theory. In *Proceedings of ACM SIGIR* (1993), pp. 270–279.

[78] Pejtersen, A. M. A library system for information retrieval based on a cognitive analysis and supported by an icon-based interface. In *Proceedings of ACM SIGIR* (1989), pp. 40–47.

[79] Rennison, Earl. Galaxy of news: An approach to visualizing and understanding expansive news landscapes. In *Proceedings of UIST 94, ACM Symposium on User Interface Software and Technology* (1994), pp. 3–12.

[80] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., and Gatford, M. Okapi at TREC-3. In *Third Text REtrieval Conference (TREC-3)* (1995), Donna Harman and Ellen Voorhees, Eds., NIST.

[81] Rocchio, Jr., J. J. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, Gerard Salton, Ed. Prentice-Hall, Inc., 1971, pp. 313–323.

[82] Rose, D. E., Mander, R., Oren, T., Ponceleon, D. B., Salomon, G., and Wong, Y. Y. Content awareness in a file system interface: implementing the 'pile' metaphor for organizing information. In *Proceedings of ACM SIGIR* (1993), pp. 260–269.

[83] Rushall, D., and Ilgen, M. D. DEPICT: Documents evaluated as PICTures: Visualizing information using context vectors and self organizing maps. In *Proceedings of IEEE Information Visualization* (1996), pp. 100–107.

[84] Salton, Gerard. *The SMART Retrieval System.* Prentice-Hall, 1971.

[85] Salton, Gerard. *Automatic Text Processing.* Addison-Wesley, 1989.

[86] Salton, Gerard, and Buckley, Cris. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science 41* (1990), 288–297.

[87] Schapire, Robert E., Singer, Yoram, and Singhal, Amit. Boosting and rocchio applied to text filtering. In *Proceedings of ACM SIGIR* (1998), pp. 215–223.

[88] Schütze, Hinrich, Hull, David A., and Pedersen, Jan O. A comparison of classifiers and document representations for the routing problem. In *Proceedings of ACM SIGIR* (1995), pp. 229–237.

[89] Sebrechts, Marc M., Cugini, John V., Laskowski, Sharon J., Vasilakis, Joanna, and Miller, Michael S. Visualization of search results: a comparative evaluation of text, 2d, and 3d interfaces. In *Proceedings of ACM SIGIR* (1999), pp. 3–10.

[90] Shavlik, Jude, Calcari, Susan, Eliassi-Rad, Tina, and Solock, Jack. An instructable, adaptive interface for discovering and monitoring information on the world-wide web. In *Proceedings of the 1999 international conference on Intelligent user interfaces* (1999), pp. 157–160.

[91] Silverstein, C., Henzinger, M., Marais, H., and Moricz, M. Analysis of a very large AltaVista query log. Tech. Rep. #1998-14, SRC, 1998. http://gatekeeper.dec.com/pub/DEC/SRC/technical-notes/abstracts/src-tn-%1998-014.html.

[92] Song, Min. Bibliomapper: A cluster-based information visualization technique. In *Proceedings of IEEE Information Visualization* (1998), pp. 130–136.

[93] Spink, A., and Losee, R. M. Feeback in information retrieval. *Annual Review of Information Science and Technology 31* (1996), 33–78.

[94] Su, Louise T. Evaluation measures for interactive information retrieval. *Information Processing and Management 28*, 4 (1992), 503–516.

[95] Sutton, Richard S., and Barto, Andrew G. *Reinforcement learning: an introduction*. The MIT Press, 1998.

[96] Swan, Russell, and Allan, James. Aspect windows, 3-d visualizations, and indirect comparisons of information retrieval systems. In *Proceedings of ACM SIGIR* (1998), pp. 173–181.

[97] Tague, Jean, and Schultz, Ryan. Evaluation of the user interface in an information retrieval system: a model. *Information Processing and Management 25*, 4 (1989), 377–389.

[98] Tombros, Anastasios, and Sanderson, Mark. Advantages of query biased summaries in information retrieval. In *Proceedings of ACM SIGIR* (1998), pp. 2–10.

[99] Turtle, Howard, and Croft, W. Bruce. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems 9*, 3 (1991), 187–222.

[100] van Rijsbergen, C. J. *Information Retrieval*. Butterworths, London, 1979. Second edition.

[101] Voorhees, Ellen M. The cluster hypothesis revisited. In *Proceedings of ACM SIGIR* (June 1985), pp. 188–196.

[102] Ward, J. H. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association 58* (1963), 236–244.

[103] Wilkinson, Ross, and Hingston, Philip. Using the cosine measure in a neural network for document retrieval. In *Proceedings of ACM SIGIR* (1991), pp. 202–210.

[104] Willett, Peter. Recent trends in hierarchic document clustering: a critical review. *Information Processing and Management 24*, 5 (1988), 577–597.

[105] Wise, James A., Thomas, James J., Pennock, Kelly, Lantrip, David, Pottier, Marc, and Schur, Anne. Visualizing the non-visual: Spatial analysis and interaction with information from text documents. In *Proceedings of IEEE Information Visualization* (1995), pp. 51–58.

[106] Witten, Ian H., Nevill-Manning, Craig, McNaub, Roger, and Conningham, Sally Jo. A public library based on full-text retrieval. *Communications of the ACM 41*, 4 (1998), 71–75.

[107] User study URL. `http://toowoomba.cs.umass.edu/~leouski/SE/`.

[108] Xu, Jinxi, and Croft, W. Bruce. Querying expansion using local and global document analysis. In *Proceedings of ACM SIGIR* (1996), pp. 4–11.

[109] Zamir, Oren, and Etzioni, Oren. Web document clustering: a feasibility demonstration. In *Proceedings of ACM SIGIR* (1998), pp. 46–54.