This is to certify that the

dissertation entitled

PERSONALIZED INFORMATION DISCOVERY
FROM UNSTRUCTURED TEXT

presented by

Marilyn Wulfekuhler

has been accepted towards fulfillment
of the requirements for

Doctoral degree in Computer Science & Engineering

Major professor

Date 8/18/98

Personalized Information Discovery from Unstructured Text

By

Marilyn Wulfekuhler

A Dissertation

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Department of Computer Science

July 31, 1998

Abstract

Personalized Information Discovery from Unstructured Text

By

Marilyn Wulfekuhler

We have developed a technique that finds conceptually related groups of words which represent the underlying content of user designated text examples. These word groups characterize the content of the documents at an intermediate conceptual level that is between full understanding and simple statistical processing of individual words. A number of user's needs for finding and organizing information can be met by using this intermediate conceptual level, beyond current methods which either rely on individual words to characterize documents, or methods that rely on a global, static comprehensive knowledge base.

Given sets of examples of unstructured text which a user has designated as interesting, we can automatically find conceptually related groups of words from them which represent underlying building blocks of the content of the documents. We find these word groups using standard statistical clustering techniques without the use of complicated natural language processing techniques, and without the need for

explicit external knowledge. We require only a relatively small number of example documents.

Both automatic document content search and automatic document classification have long been goals in information retrieval which have had only limited success. We conjecture that success has been limited because in the case of classification, the traditional focus has been to achieve a global, static, comprehensive scheme which is all things to all people and all documents. In the case of search, traditional methods have focused on documents and information needs at an individual word level; when words are considered in isolation, the problems of polysemy (identical words with different meanings) and synonymy (different words with identical meanings) limit the success of the search.

By using an intermediate conceptual level, we can arrive at *personalized* context sensitive organizational schemes, and provide a more useful basis for search than that provided by representations based on isolated words.

To my family, Ted, Daniel, Erika

ACKNOWLEDGMENTS

I would first like to thank my advisor, Bill Punch, for his extraordinary support and patience. I could not have asked for a better advisor, and I was lucky to have the privilege of working with him. Thanks also go to the other members of my guidance committee: Anil Jain, who was always interested in my academic success and always expected excellence, Nicholas Altiero, who carefully reviewed my work and offered helpful suggestions for improvements, and Mike Moch, who kept an eye on the practical application of my research.

I gratefully acknowledge financial support from the Dean's Distinguished Fellowship program and NEM–Online. Without the financial assistance they provided, it is likely that I would have given up on a Ph.D. years ago.

I would also like to thank my fellow graduate students when I was in the PRIP lab, including Chitra Dorai, Dan Swets, Sharath Pankanti, and Sally Howden. These people were not only good colleagues, but good friends. Also from my days in the PRIP lab, I would like to thank George Stockman and the late Richard Dubes for starting me down the path into research.

When I found a new home in the GARAGe, I also found more good colleagues

and friends, including Eric Goodman, Owen Matthews, Terry Dexter, Jason Greanya, Gang Wang, Victor Miagkikh, Eman El-Sheikh, Oleg Lukibanov, and Iliana Martinez-Bermoudes. These individuals contributed to a stimulating and enjoyable environment. I would like to single out GARAGe dwellers Boris Gelfand, Arnold Patton, and Mike Raymer for special recognition, for listening to my brainstorming ideas, for their willingness to give suggestions and opinions, as well as for just being there with moral support. Also deserving special thanks are Hugh Smith, Donna Wicks, Darren Meahl, Marie-Pierre Dubuisson Jolly, and Qian Huang, who kept me going through particularly black times.

I would like to thank my parents, my sister Margaret and my brother Carl for their unwavering encouragement and support. I would also like to thank my children Erika and Daniel for putting up with my late nights and time spent in the lab. And finally, I want to thank my husband Ted for his understanding, patience, commitment, and support. This degree would not have been possible without his help.

TABLE OF CONTENTS

LIST OF TABLES x

LIST OF FIGURES xi

1 Introduction 1
1.1 Motivation . . . 4
1.1.1 Information and Knowledge . . . 4
1.1.2 A new domain – the Internet and World Wide Web . . . 5
1.2 Representation of Information . . . 7
1.2.1 Conceptual level representation . . . 7
1.3 Information Discovery in a library vs. the Internet . . . 11

2 Literature Review 18
2.1 Traditional Information Retrieval . . . 19
2.1.1 Word Level Representation . . . 19
2.1.2 Feature Selection and Extraction . . . 21
2.1.3 Query Modification . . . 24
2.1.4 Evaluation of Search Effectiveness . . . 28
2.1.5 Traditional IR Search vs. On–Line Search . . . 30
2.1.6 Search on the World Wide Web . . . 31
2.2 Organization of Textual Information . . . 32
2.2.1 Global collection representation . . . 33
2.2.2 Subject Catalogs . . . 34
2.2.3 Document Classification . . . 35
2.2.4 Document Clustering . . . 41
2.3 Summary . . . 48

3 Finding Concepts 49
3.1 Preprocessing . . . 51
3.1.1 Stemming . . . 52
3.2 Forming the Pattern Matrix . . . 53
3.3 Finding Important Features . . . 54
3.3.1 Feature Selection . . . 55
3.3.2 Feature Extraction . . . 59
3.4 Transposing the Matrix: Clustering Words Instead of Documents . . . 64
3.4.1 Choosing a value for K . . . 65
3.4.2 Distance Measures . . . 66
3.4.3 Normalization for document length . . . 69

3.4.4 Combining runs into fuzzy sets 70
3.5 Summary 75

4 Experiments and Results **76**
4.1 Sample Data Sets 76
4.2 Word Clustering Results 83
4.2.1 Statistically Random Data Set 83
4.2.2 Random Seeded Data Set 85
4.2.3 Reuters Data Set 91
4.2.4 NEM-Online Data Set 93
4.3 Cluster Usefulness 95
4.3.1 Classification Accuracy 95
4.3.2 Search for Similar Documents 96
4.3.3 Personalized Document Organization 100
4.3.4 Conclusions 103

5 Summary and Future Work **105**
5.1 Summary 105
5.2 Future Work 108
5.2.1 Characterization of Training Set 108
5.2.2 Hierarchical Subclustering 109
5.2.3 Other methods for determining fuzzy set membership 110
5.2.4 Applications for conceptual representation 110
5.2.5 Incorporate knowledge in representation 110

List of Tables

3.1 Sample of the partial pattern matrix. 54

4.1 Seed words from four Reuters categories 80
4.2 Sample Reuters data set categories 81
4.3 Sample NEM Online data set categories 82
4.4 Summary of the training data sets: Statistically Random (StR), Seeded Random (SeR) at 1%, 5%, 10% seed words, Reuters articles (RA), and NEM–Online (NO). 83
4.5 Three smallest, strongest clusters from 5% random seeded document set 89
4.6 Two clusters containing seed words from 10% random seeded document set 90
4.7 The four clusters from actual Reuters articles whose strongest member was greater than .4. Words with strength $< .7$ appear in the clusters, but are not shown here. 92
4.8 The four clusters from NEM–Online documents. 94
4.9 Occurrences of cluster words in all Reuters documents of the four categories 98
4.10 Precision and Recall of single word queries 99

List of Figures

1.1 General Overview of the System 10

3.1 Four documents in 2 dimensions 67

4.1 Sizes of $K-1$ smallest clusters in all data sets 84
4.2 Maximum Precision and Recall for Reuters cluster 1 in the gas category . 98
4.3 Maximum Precision and Recall for Reuters cluster 2 in the grain category 99

Chapter 1

Introduction

We have developed a technique that finds conceptually related groups of words which represent the underlying content of user designated text examples. These word groups characterize the content of the documents at an intermediate conceptual level that is between full understanding and simple statistical processing of individual words. A number of user's needs for finding and organizing information can be met by using this intermediate conceptual level of characterization, beyond current methods which either rely on individual words to characterize documents, or methods that rely on a global, static, comprehensive knowledge base.

We can concisely state the problem as follows:

> Given a set of n textual documents that a user has identified as "interesting", containing d total words (the features), find some subsets of d which represent *conceptual features* which distinguish these documents from others, and which represent concepts contained in the n documents.

The subsets of related words are found using standard statistical clustering techniques without the use of complicated natural language processing, and without the need for external knowledge. We require only a relatively small number of example documents; we do not try to build these conceptual features based on a large corpus of text; we use only documents which represent the user's focus of attention at the time.

We can use these conceptual features and the topics they suggest as the characterization of textual information in order to:

- automatically search for unknown documents on similar topics
- organize collections of documents tailored to personal preference
- dynamically classify document collections with *topical proximity*

These tasks are less fruitful under current schemes because traditional methods represent information either by using individual words in documents, which is too narrow a view, or by using a globally imposed topic categorization, which is too broad. Representing information at an intermediate conceptual level enables a personalized, dynamic view of textual information, allowing the above tasks to be performed more effectively. This is the major contribution of this thesis.

Assumptions we make are as follows:

1. A user has collected several documents from each of several categories of interest so that the document collection contains both commonality (as defined by a user) between documents in the same category, and diversity among categories.

2. The commonality consists of related, low–level concepts which likely share a

specialized vocabulary rather than meta concepts (e.g., a collection of funny stories).

3. The documents are not labeled by category.

4. The number of documents n is smaller than the number of distinct words d.

5. A user may generate new conceptual features at any time, as interests or document examples change.

Assumption 4 states that $n < d$. If there are more documents than distinct words, that is, $n > d$, the document collection can be broken into subsets such that the commonality and diversity requirements are met. The reason for the restriction on the size of the document sets will be further explained in subsequent sections.

Assumption 5 implies that total processing time should be short enough to allow user feedback as quickly as possible. While real-time processing is not yet possible, it should not take more than a few hours (e.g., over lunch or overnight) to generate conceptual features and use them for search or organization.

In this chapter we motivate the problem, describe how it has been handled historically in the context of library science, and explain why traditional methods don't work well in the new domain of the Internet. Chapter 2 reviews previous related work and current approaches to information discovery on the Internet. Chapter 3 describes our system for extracting conceptual features which we use for representing information at an intermediate conceptual level in a personal, dynamic way. In this chapter we also describe our attempts to achieve personal categorization using word-level features, and explain why categorization using this sort of representation is doomed to

failure. Chapter 4 gives detailed results of our experiments using 4 different data sets. It then gives preliminary results of using the conceptual level features for automatic search and document organization. Chapter 5 gives a summary and outlines future work to be done in the area of automatic information discovery.

1.1 Motivation

As the information age proceeds, we are deluged with more and more information, and it is becoming increasingly difficult to manage it. An individual can not and does not want to understand the organization of all available information, only that which is of current interest to him or her. We need to have an easily derived representation of information that conveys meaning in a given context which represents a user's current focus of attention.

1.1.1 Information and Knowledge

Humans are unique in the animal kingdom in that they do not rely solely on experience or a direct teacher to gain information. Instead, they can utilize an amassed body of knowledge which has been acquired and stored by others. Significant among sources of all information are *textual sources*, for which we have developed well established practices of access. Some information needs are specific, (e.g., I need to know the maximum recorded length of a blue whale), while others are general (I want to learn about blue whales). Representation of information encompasses both these needs, and information sources which fulfill needs. The traditional resource for

acquiring knowledge from textual information is a library, and methods and solutions for meeting information needs have been developed in that context.

1.1.2 A new domain – the Internet and World Wide Web

The explosion of the Internet and the World Wide Web in the past four years has created an unprecedented amount of readily and widely available textual information. The current approaches to discovering relevant information are typically patterned after methods and techniques from the information retrieval field, where the problem of automatically finding information from textual sources has been well studied in the context of library science. However, there are fundamental differences in the basic assumptions of search and classification of text in library contexts which make traditional methods inadequate when seeking information from the Internet.

Specifically, published works in a library are collected under well defined rules and controls, using well established techniques. The tasks of categorizing, indexing, and representing information are performed by professional humans. With the rise of the Internet as a source of textual information, however, anyone can publish information, and a document is under complete control of the author, who is most likely to be untrained in library science. Library infrastructure and controls are absent in the Internet, providing unique challenges to information discovery. There have been attempts to automate some of the tasks performed by professional librarians, both in library collections and on the Internet. Here we briefly describe traditional approaches in broad terms; more detailed explanation is given in section 1.3 and in

chapter 2.

Both automatic document content search and automatic document classification have long been goals in information retrieval though researchers have achieved only limited success.

In the case of classification, the traditional focus has been to find a global, static, comprehensive categorization which is all things to all people and all documents. In the case of search, traditional methods have focused on documents and information needs at an individual word level; when words are considered in isolation, the problems of polysemy (identical words with different meanings) and synonymy (different words with identical meanings) limit the success of the search.

Rather than attempt to classify or understand the universe, we consider only those concepts and categories that are important at a given time to a given user, and are therefore able to dynamically provide a reasonable approximation of content components of a document *in a given context.* In our approach we examine small sets of documents from which we derive groups of related words (related in the sense that they have similar patterns of usage across the example documents). Using groups of words together rather than in isolation alleviates the polysemy and synonymy problems because the context of the other words in the set can disambiguate polysemic words, and can augment a single word with its synonyms.

1.2 Representation of Information

Textual documents are an abstraction of the information they contain. Representation of documents, in turn, is a further abstraction, preserving the expression of information as much as possible while providing a computationally feasible way to analyze, categorize, and search.

There are two main approaches to representing documents: One extreme considers a document as being composed of individual words or phrases. Though the underlying semantics can not in general be obtained automatically, the idea is that documents with similar semantics use similar words, thus allowing a word search match. The other extreme consists of looking at a document as a whole, deciding what one thing it is about, and assigning it a label in a global information categorization. Clearly there is room for a middle ground, where documents are seen as being composed of individual coherent topics. This is the approach we take.

1.2.1 Conceptual level representation

Rather than using either a word level representation or a global topical representation for textual sources, we consider documents at an intermediate conceptual level.

Suppose a user has a collection of n interesting documents logically grouped together in classes either explicitly (a user's bookmark file, or email folders, for example), or implicitly in the user's mind, according to the individual's ideas of what constitutes a category. Two people may have the same collection of documents, but classify them differently. For example, one user may have saved a given page from a

company which does research on software agents because of her interest in one of its products which facilitates Internet commerce. They would (explicitly or implicitly) group it with other pages about similar products, while another user may have saved the same page because of interest in obtaining a job there, and grouped it with other job leads.

Note that the user need not label the groupings; but when the whole collection is examined, it is apparent to the user that a certain subset of the documents refers to certain conceptual categories. The user–defined grouping is one that makes sense to that user in the context of the other documents in the set. This means that the underlying *conceptual features* which determine the user's categories will vary with each individual document collection, and with each user. The set of documents a user has saved provides *implicit context* in determining the semantic concepts that were used. In trying to extract these conceptual features, it is not enough to merely find features that reproduce the user's classification of their small example document set. A number of traditional feature selection techniques from individual words can reproduce the classification with a high degree of accuracy, as will be described later. However, this is not enough, since we want features that are at a higher conceptual level than single words, and which suggest the *concepts* that underly the user's categories. We can find these conceptual features as long as the underlying concepts share a common vocabulary; meta concepts such as "humorous pages", or "ugly color schemes" will not be detected.

We will enable personalized information discovery from unstructured text by first finding conceptual features from example documents. These features consist of groups

of semantically related words (concept clusters) with a value associated with each word according to its "strength" in the concept. The user may manually refine and/or label the concept clusters to better define their personal ideas of the topic, but intervention is not required.

Note that these concept clusters are different from any classes that the user may have defined in their example documents, and are also different from the notion of classes provided by global subject catalogs such as Yahoo!. The word groups are generated *dynamically* according to the raw features from the example document set, and will change as the example document set changes.

Representing documents at an intermediate level through our conceptually related word groups is useful in a number of ways. We can perform automatic searches on behalf of a user based on topics they are interested in, rather than forcing them to construct queries composed of individual words which are iteratively submitted and refined. We can also improve the presentation of search results to the user by classifying them according to the concepts the user has already indicated interest in. Rather than merely presenting a linear list of ranked results as most search services do, we can present documents of potential interest in a way that allows a user to browse them with the expectation that physical proximity of the documents in the presentation corresponds to their topical proximity.

We initially examine single words contained in documents, however, once our word groups are identified, each group as a whole has an evident semantic meaning which does not rely on a universally defined knowledge base of meaning built by human categorization, and which is more powerful than considering words in isolation. The

word groups also alleviate the problems of polysemy and synonymy by providing a context for individual words.

Providing a system to automatically classify/organize documents according to semantic content as defined by the user, and automatically searching for previously unseen documents with similar semantic content are further contributions of this thesis.

A general overview of the system is given in Figure 1.1, and will be described in detail in Chapters 3 and 4.

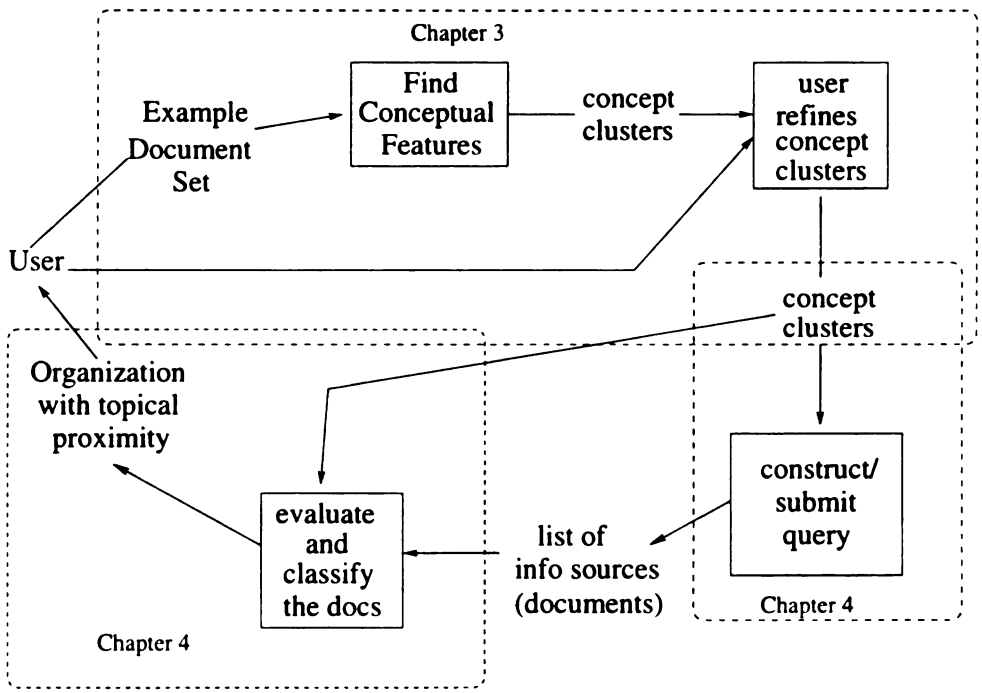

Figure 1.1: General Overview of the System

1.3 Information Discovery in a library vs. the Internet

There are two distinct but related activities which facilitate information discovery from textual sources (for example, the WWW, newsgroups, or a traditional library): *searching* and *browsing*. Search consists of submitting a query to a predefined index, relying on a word level representation of documents, while browsing consists of briefly examining several items, relying on a global concept categorization. While both activities can arise from a specific or a general information need, search is typically for specific information, while browsing is typically for general information.

In a traditional library, search is facilitated by the card catalog, which has an established indexing scheme which maps individual query words (including keywords, author, title, etc) to a document. The document has a unique identifier (Dewey Decimal System number [2] or Library of Congress Subject Heading [3]) which enables the user to precisely locate the document in the library. The indexing scheme and identifier also facilitate browsing, since the identifiers are constructed such that items about similar topics are intended to be in close physical proximity to each other. Thus, once a user locates a single item of interest, others on the same topic will be physically nearby. The underlying assumption of this scheme is that a given document identifier has a well defined mapping to the semantic content or subject matter of the document. Likewise, there is a well understood inverse mapping from subject matter to identifier. When new documents are published, the labor intensive process of determining the details of this mapping is performed by specially trained people.

On the Internet, every document does have a unique identifier in the form of a Universal Resource Locator (URL), but there is no well-defined relation between topic and location. The URL of a web document reflects wherever the author happened to put it. There is no sense of *topic proximity*: items on the same topic can be widely scattered across the world. Users browse by following hyperlinks, which presumably go from a given page to a related topic, but the relevance of the relationship may only be known to the page author.

A further difficulty on the Internet is that there is no well-defined notion of a "document". A document on the World Wide Web is usually thought of as a single item with a distinct URL, but may logically be only a part of a document (for example, a single section from a scientific paper), or logically several documents (for example, a Usenet newsgroup digest). Whenever we refer to an on-line "document", we will mean a data item that has a unique URL as an identifier. This may include book chapters, directory lists, a single email message, digests, etc.

Current search of on-line sources consist of engines which are based on traditional information retrieval techniques, with some modifications, analogous to the library card catalog search. A user submits a *query* consisting of a Boolean expression of keywords to a service and is returned a set of documents which match the query.

In a library card catalog, a user can search for a document by title, author, or keyword. Searching on the Internet is based only on keywords, not title or author. For example, in a search for C.J. van Rijsbergen's book "Information Retrieval", in the Michigan State University library's MAGIC card catalog system, a query specifically for the title as "t=Information Retrieval" results in three items with that title, all

with the Library of Congress subject heading of Z699. Armed with this information, the user could go to the shelf in the library containing "Z699" call numbers and browse for further resources related to the topic of information retrieval, or pinpoint the one book of three they had in mind.

However, to find an on–line version of the same book on the web [1], there is no uniform way to search for a title, and if the author's name (or even its spelling) has been forgotten, the user is forced to search for the title as a keyword. Submitting the query *information retrieval* to AltaVista, for example, returns the URLs of 30,000 documents which each must be examined to find the desired reference (the correct reference did not appear in the first 40 pages listed). Some search services do offer the capability to search specifically for a Web page's title, but since web pages do not conform to the standard notion of a single volume of work, there is no guarantee that the title of a Web page containing the book is the title of the book. (Titles of the Web pages comprising that book are in fact unspecified and so default to the file names "Preface.html", "Ch.1.html", "Ch.2.html", etc.)

Another important difference between library search and on–line search lies in the construction of the keyword index. While library card catalogs are hand constructed with carefully selected keywords, major web search engines such as Lycos [6], AltaVista [7], WebCrawler [8], and Open Text [9] employ software robots [10] (also called spiders) which traverse the web, read each document, and automatically create an index of the documents they find. The index is an attempt to somehow characterize a document without storing its full text. Each service makes its own

[1] its URL is http://www.dcs.gla.ac.uk/Keith/Preface.html

decisions about how much of a document to read, which words from the document belong in the index, and the weight or importance of each indexing term. Although there is a facility for associating keywords with Web documents (using META tags within HTML documents), the responsibility for specifying them lies with the document author. These META tags do not affect the information displayed to a user, so the author may choose keywords poorly, may deliberately choose wrong keywords to fool indexing robots, or may simply not bother to specify any at all. Since the software robots can not rely on the "keywords" given by web page authors, and since they do not have the intelligence of a professional librarian, the most comprehensive search engines treat *every* word in an on–line document as a potential keyword, which results in large numbers of irrelevant documents being returned for a query, simply because the query word occurred somewhere in the document, not because the query word was an important keyword of that document.

Since various web search engines differ in their indexing criteria, coverage, and recency, identical queries to different services return different results. Query format also varies with the particular implementation, and most users have difficulty in formulating optimal queries for all the search engines. If the user is skilled enough to formulate an appropriate query, most search engines retrieve pages with adequate *recall* (the percent of the relevant pages retrieved among all possible relevant pages), but with poor *precision* (the ratio of relevant pages to the total number of pages retrieved), because of the keyword selection problem described above. The important point is that searches are conducted and matching documents are found based on an individual word level of representation, rather than an idea or topic level.

For example, a naive user looking for information on legal issues involved in running a small business might try the query `legal small business`, which to AltaVista means "give me all the documents which contain any of the words *legal*, *small*, or *business*. This query returns over a million documents (1,813,308 contained *legal*, 4,133,148 contained *small*, and the 8,805,231 containing *business* were ignored by AltaVista). A more sophisticated user who is familiar with AltaVista's query syntax might try the query `+legal +"small business"`, which means "give me all the documents which contain the word *legal* AND the phrase *small business*. This query returns around 10,000 documents, still an overwhelming amount for a user to sift through, and they are not relevant or precise.

The results of these kinds of searches are typically presented as a list of URLs matching the documents along with their first 20–30 words. The list is usually ranked according to the search engine's notion of relevance to the query, which may give higher rank to documents which contain the search terms in the title, documents which contain the search terms more times than others, or other schemes which are usually not made known to the public. In any case, there is no topical proximity in the returned list, and the user must search linearly through the list, possibly fetching the full document, in order to determine if any of the listed documents are of interest.

In contrast to keyword based representations, global categorization schemes create *directories* (hierarchical categories), and are an attempt to provide some topical proximity in Internet documents, analogous to the Dewey Decimal System or Library of Congress Subject Headings in libraries. Yahoo! [4] is the largest and most well known Internet directory, but others also exist, sometimes alongside search engines

(e.g, Infoseek [5] and Lycos [6]). To construct a directory, individual documents are submitted or collected by hand, then assigned to a category in the hierarchical organization of topics created by the directory designers. Browsing is facilitated by providing the hierarchical topic organization. For example, to find information on karate, it is easy to start at Yahoo!'s top level, and successively travel down the hierarchy to *Recreation and Sports*, then to *Sports*, *Martial Arts*, and finally *Karate*, where there are 101 documents listed.

However, not all topics are as clear cut. For example, to search for a down-loadable version of the publicly available stemming code from the textbook "Information Retrieval: Data Structures and Algorithms" by Frakes and Baeza–Yates [11], one might guess to start at the top level Yahoo! category of *Computers and Internet*, then the sub category *Software*, then perhaps *Archives*. In any case, trying to guess the categories from this point leads to frustration, since the code in question is nowhere to be found within the *Computers and Internet* sub–hierarchy. It is found in Yahoo!, but under the top level category *Reference*, then *Libraries*, *Library and Information Science*, and finally, *Information Retrieval*. Successful navigation of the topic hierarchy requires that the user have the same notion of topic organization used by the designers, not the users' personal organization.

Another problem with a global categorization is that the number of documents listed in a subject directory is necessarily much smaller than that in search engine indices because of the human categorization bottleneck. There have been attempts at automatic classification of web documents (e.g., [12], [13]), but these have had limited success.

Rather than relying on user-specified queries of keyword based searches, or global subject categorizations, we will instead use our automatically determined features which describe the content of a document at a higher level than individual words, according to an individual user's interests. We then use these higher level features to seek out additional information and organize other documents (which may be candidate interesting documents resulting from a search, or merely previously unclassified documents we already had on hand) in a topically related browsable form. This is an improvement over current methods for information discovery on the Internet.

Chapter 2

Literature Review

Information discovery from textual sources is a fairly mature field in the context of libraries, but has received much renewed interest recently because of the vast new amounts available from on–line sources such as the World Wide Web. Automatic methods for information discovery in traditional settings make up the field of Information Retrieval (IR),where the focus is typically on *search* for documents which meet a need that is expressed in a *query*. Many of the current methods of discovering information from on–line sources have roots in traditional IR techniques, so we begin this chapter with an overview of IR and the most widely used model of information, Salton's vector space model [14] and its variants and assumptions. With this background, we can also examine current methods for on–line search. The other major activity in information discovery, *browsing*, is dependent on the specific organization of text collections, so we next review various methods of text collection organization, and conclude with current techniques for browsing assistance without an explicit organization.

2.1 Traditional Information Retrieval

2.1.1 Word Level Representation

Information Retrieval [14, 15, 11] is a mature field which grew out of a need for automatically searching large collections of text such as those found in a library. In order to meet a user's need for information, a *query* is given, and pointers to documents that are relevant to that query are returned to the user. Representation of documents, representation of queries, and determination of relevance are important issues in IR.

The most widely used model for query–document matching is Salton's vector space model [14], where each document (pattern) is represented as a vector of words (features). In this approach, n words form an n–dimensional feature space, and each document lies somewhere in that space. A query for information is composed of words, so it can also be represented as a vector in that same feature space. Matching a query to documents then consists of plotting the query in the space, and returning the documents which lie closest to the query according to some distance measure. The values for each vector element can range from binary (where a 1 indicates occurrence of the word in the document, a 0 indicates absence), to values derived from the frequency of the word in the document, to other more complex weighting schemes. Choice of a distance measure is coupled with the representation of the feature vectors. Possibilities include Euclidean distance, cosine coefficient, Dice's coefficient, Jaccard's coefficient, etc. [15]

For example, a popular way to form values for the components in the feature vector

is to use the *term frequency/inverse document frequency* (TF/IDF) measure [14], [15] for weighting each term. [1] Term frequency reflects the number of times a given word occurs in a single document, and document frequency refers to the percentage of documents from the entire collection in which the term appears. The idea behind the TF/IDF measure is that words that occur often in a document, but rarely across the entire document set, are important words and should receive a higher value (weight) in the feature vector.

The choice of a distance measure between documents, or between documents and queries, can imply an assumption about the importance (or lack thereof) of the length of a document. Most of the distance measures used, including the most popular cosine coefficient, automatically normalize for document length [15]. For library collections, where the types of documents being indexed are fairly homogeneous in form, it is reasonable to assume that terms in long documents should not be weighted more heavily than terms in short documents. However, Web documents can range from simple lists of URLs to complete texts. The assumption that long and short documents should be equally treated for indexing and representation purposes may not be a good assumption for Web documents, as we will discuss more fully later.

Many studies have been done on effective normalization schemes (e.g., [16]), but all of these have been with document collections (TREC, Reuters) whose members do not vary much with respect to style, unlike Web documents.

[1] A *term* is technically different from a *word* in the information retrieval literature, reflecting the fact that the original word may have been stemmed or changed to some canonical form, but we will use *term* and *word* interchangeably, and will note stemming or other operations when appropriate.

2.1.2 Feature Selection and Extraction

When considering documents as patterns and the words in documents as features, it obviously does not make sense to consider all words as equally important. We know that not all words in a document convey information, and that some words convey more conceptual content than others, so feature selection and extraction is an important aspect of document representation.

It should be noted that sometimes all words in a document are not even eligible for feature consideration. Due to computational limitations, many systems in the past only used keywords chosen by humans, or words from only the title and abstract.

As early as 1957, Luhn suggested that the most frequently appearing words in a document collection, as well as the least frequently appearing words, are not as significant [17]. Selecting features based on the "Luhn cutoffs" is still a popular method of feature selection, although there is no well established way to specify the thresholds for the cutoffs.

Related to the idea of most frequent words not having high information content is the notion of a negative dictionary, or *stoplist* [18]. Words such as "and", "of", "the" should be eliminated from consideration as features, no matter what their frequency.

Another operation which reduces features is *stemming* [19]. Stemming is the process which reduces a word to its stem (e.g., "computer" and "computing" both become "comput"). Stemming explicitly acknowledges the fact that distinct original features are not independent.

Bookstein et. al. [20] attempted to identify content bearing words based not only

on frequency, but also based on the sequential occurrence pattern of the words. The hypothesis is that words which tend to appear in "clumps" throughout the document convey more information than words which are distributed randomly. This method uses the key idea that a given word may be important in one set of documents, but irrelevant in another.

One method of both reducing the feature space and eliminating feature interdependence is *Latent Semantic Indexing* [21]. Here the original pattern matrix is converted into an orthogonal one, which can be truncated according to those vectors which correspond to the smallest eigenvalues. This is basically the principal components projection. It is claimed that the newly derived feature vectors contain "latent semantics" of the documents - semantics that can not be described explicitly, but are present as a linear combination of the original features. We attempted a similar approach, as will be described in the next chapter, but while latent semantic indexing may be useful for document classification accuracy, it is ineffective for search on large dynamic data collections, such as the World Wide Web.

An extension to the basic vector space model which is related to the idea of latent semantic indexing is the Generalized Vector Space Model (GVSM) [22]. GVSM, recognizing that one of the limitations of the Vector Space Model is that terms are not independent (especially when they have multiple meanings), considers each word to be independent of the others, unless two words both index the same document. The method then creates orthonormal basis vectors from the document–term frequency matrix, and derives a similarity measure incorporating these basis vectors. It was reported that precision was improved over the basic VSM model, but using linear

combinations of the original terms does not convey conceptual meaning of the documents. This model was proposed during a time when documents were represented by a small set of keywords.

Another attempt to capture semantics in the representation of documents for search is the Semantic Vector Space Model [23]. This model also recognized that the basic vector space model, as a set of words, does not capture any contextual information about a word's usage in a given document. The semantic vector space model extends the basic vector space model by using a heuristic natural language parsing technique which assesses the probability that a word in a given sentence plays one of a set of *thematic roles* as defined by linguists. A given term t in a document can play one of m thematic roles. Probabilities for each of the possible roles are computed for k occurrences in a single document, which are then combined linearly into a single length m term vector, V_T. A single document, then, is represented, not as a weighted term vector (with n terms), but as a $n \times (m+1)$ matrix, where the rows of the matrix represent a vector for each term in the document. These row vectors contain first, the original term weight from the traditional vector space model, then the term vector V_T which represents the roles that term might take in the document. In other words, the first column of the matrix is the traditional weights from the vector space model. One issue that arises in using this model is the problem of computing similarity between matrices. It remains to be seen whether this fairly recent model will prove useful.

2.1.3 Query Modification

When a user has an information need, it is usually specified in the form of a query - a free text or boolean expression consisting of keywords. The query is represented as a vector of its component words, then plotted in the same vector space as the documents are, so that documents "near" the query can be returned as the most likely information relevant to the need. This is a simple and mathematically easy way to match information sources with information needs, but queries typically have many fewer terms than even the shortest documents, and while the mismatch in length is not too much of a problem when documents are represented by relatively few keywords, the length difference is greatly magnified for World Wide Web documents, which are usually indexed by their full text, and for which typical user queries consist of only two words [24]. The mismatch which results when a document is represented by too many terms, and a query is represented by too few terms, causes queries to match too many documents, and leads to the low precision characterized by Web searching.

It has long been recognized that user generated queries are imperfect indicators of information needs [25]. One technique which addresses this problem is known as *query modification* (also known as *relevance feedback*), which weights the terms in a query according to the user's evaluation of documents returned. In the first iteration, when the documents resulting from a user's query are returned, the user labels individual documents as relevant or irrelevant. Then the distribution of the query terms in these two document sets are analyzed, and the query terms are assigned weights accordingly. For example, a query word which occurs frequently in the set of relevant

documents but infrequently in irrelevant documents receives a higher weight, while terms occurring frequently in irrelevant documents and infrequently in relevant ones will receive lower weight. The newly weighted query is then submitted to the system, which returns another set of documents, which the user again evaluates as relevant or non relevant. There may be many iterations, all of which require the user to evaluate every document returned at each stage, which becomes tedious.

Another technique for more closely matching a user–defined query to actual information needs is *query expansion*, which modifies the query by adding additional words from a thesaurus.[2] The thesaurus may be constructed by hand, which is very labor intensive and time consuming, or it may be constructed automatically.

Word clustering is an appealing idea to automatically construct a thesaurus, and early work using this technique was done by Sparck–Jones [26]. In that work it was pointed out that though the intent at the time was to improve recall in retrieval by adding words not included in the original query, use of an automatically constructed thesaurus could improve precision as well, by disambiguating word senses. Words from the entire document collection are clustered to obtain thesaurus classes. This early work was only applied to small (by today's standards) data sets, and only marginal improvements in recall and precision were realized, as has been the case with many extensions to this approach since the early work [27] [28] [29]. A global approach to clustering the words identifies words that have similar co–occurrences across the entire corpus of documents, but it does not take into account the idea

[2]A thesaurus in information retrieval is not like the lay man's idea of a thesaurus which contains only synonyms. An IR thesaurus entry for a given word may contain synonyms, broader terms, narrower terms, or related terms.

that different information needs will have different thesaurus classes. Global methods do not capture the *context* of a meaningful subset of documents which reflect the information need.

In contrast to global methods, there have been *local* methods of automatic thesaurus construction described in early literature [30, 31]. In these approaches, only the top ranked documents returned after the results of the initial query were analyzed for local feedback. Based on analysis of these returned documents, queries were both expanded and re-weighted. Rather than relying on the user to determine which were relevant and which were not, these approaches assumed relevance, and adjusted the queries accordingly. A flaw in this approach is that if the n top ranked returned documents (or even a subset of them) are in fact not relevant (as is often the case in Web searching), the analysis of the words contained in these non-relevant documents will skew the context. Again, improvements in retrieval using this early approach were not promising.

More recently, Xu and Croft have proposed an approach to query expansion using global analysis methods on a local subset of documents, which they call *local context analysis* [32]. They use noun groups (rather than individual words) as concepts, which are chosen from the top ranked documents (as in the local feedback described above). However, instead of using the "best" documents as a whole to choose the concepts, they use the "best" subsets of documents, which in this case are text windows with 300 words. They report better results in terms of precision and recall using this approach than with using either global analysis or local analysis alone.

Crouch and Yang [33] describe a method for automatic thesaurus construction

which clusters *documents* rather than words in an attempt to capture context from those documents seen to be similar to each other in some way. These clusters of documents are then used as a local set on which to do analysis to construct thesaurus classes.

AltaVista's Live Topics

AltaVista has attempted to aid the user in refining their query through their experimental *Live Topics* interface to the AltaVista search results. When many documents are returned, the system gives suggestions for additional terms which may improve the query. The user can select more terms (or delete some), and the query is resubmitted. Live Topics is a form relevance feedback for generating better queries, which still must be submitted in the usual way.

With an original query consisting of the single word "clustering", the AltaVista search engine returns 20,000 documents. Live Topics suggests only 2 additional words to refine the query: *Gaba* and *Bearer*. On another try with the same query, the only two suggested words were *expatriate* and *Anderson*. These two sessions of Live Topics were of no help in narrowing down the 20,000 documents which matched "clustering".

In another Live Topics session, the original query consisting of the phrase "curse of dimensionality" resulted in about 200 matches; among the terms live topics suggested for query refinement included "clustering", "statistical", "feedforward", "hyperplane", "Gaussian", "irrelevant". When "clustering" and "irrelevant" were clicked on, and the query resubmitted to include these keywords, AltaVista returned. no matches. This experience illustrates that for some topics, Live Topics encourages

query refinement to the point of narrowing down the search so much that there are no longer any matches.

According to the documentation, Live Topics works best when the number of initial matches are 200 or more [34].

Live Topics does not consider the relevance rankings of the matches to the initial query. Rather, it analyzes *all* documents which provide a match [34]. Live Topics may not work well because it's enhanced queries are dependent on the quality of the initial set of returned documents. If many irrelevant documents are returned as a result of a query, using words from those irrelevant documents to augment the query is not appropriate. Our approach instead clusters words from documents already known to be relevant.

2.1.4 Evaluation of Search Effectiveness

Evaluation of information retrieval systems is a difficult problem [15, 35]. Even though there has been more than 20 years of research in retrieval effectiveness evaluation, nearly all evaluation measures are based on *precision* and *recall.* As mentioned in the introduction, recall is the percentage of all relevant documents which were actually retrieved, and precision is the percentage of the documents retrieved which were actually relevant. More formally,

$$recall = \frac{|relevant \cap retrieved|}{|relevant|}$$

$$precision = \frac{|relevant \cap retrieved|}{|retrieved|}$$

Although these measures and those based on them can be mathematically defined, they rely on the subjective notion of relevance. They also require a binary partition of documents into the absolute classes "relevant" and "non relevant". Therefore, comparisons of retrieval systems are often done on standard document collections where a large number of specific queries and relevance judgments have been labeled with respect to those two classes by experts.

Yao [36] suggests a performance measure which assumes documents are arranged by a weak order of relevance (which the user determines by comparing two documents at a time), leading to an ordinal scale of relevance of a number of equivalence classes of documents (rather than just two). The performance of a system is based on the distance between the user's weak ordering and the system's ordering. The traditional precision and recall measures can also be derived from the distance. The advantages of this approach are that documents can have an arbitrary number of levels of relevance, and each user may define their own personal ordering. The disadvantage of measuring system performance in this way is that the user must define their preference ordering of the entire document collection for every information need, which is clearly not practical in real systems. However, the idea of a user-defined weak ordering is a promising new development in measuring the performance of retrieval systems. In operational systems (as opposed to experimental ones), it is only the individual user who can truly determine the effectiveness of a retrieval system.

2.1.5 Traditional IR Search vs. On–Line Search

The search for specific information from on–line sources can use the traditional information retrieval model of a specific query which is then matched against a pre–existing index of documents. However, there are differences between traditional published works (where keyword selection, for example, is under the control of a publisher), and on–line information, where an author is also the publisher, and does not have the same discipline, training, or motives as print publishers. These differences create difficulties in applying the library science information retrieval model to Internet documents. Since an author can not be relied on to provide appropriate indexing information, automatic methods are used to construct massive indices of web pages. And since automatic methods are incapable of intelligently deciding which terms to index, the most comprehensive index *all* terms found in a document. This leads to many documents matching a given query, usually more than the user can easily examine. Most search engines provide a ranking of the matched documents in order of relevance. However, there is no universally accepted measure of relevance, and different search services use many different measures and algorithms in determining the score for a given document. Often, the list returned from a search contains irrelevant pages being ranked highly, while those truly relevant to a user are buried so far down the list that they are not examined.

One of the assumptions used in traditional information retrieval models is that the keywords (those used to construct the pattern vectors and which define the feature space) are a proper subset of all of the words in the documents. Usually this subset

is much smaller than the set of the total occurring words, which means that there is a somewhat controlled vocabulary. One of the challenges of automatic index building is the appropriate selection of the subset of words which comprise the feature space. Most information retrieval systems use statistical properties of the frequencies of word occurrences or co– occurrences to select the keywords.

Another assumption in information retrieval systems is that the entire collection of documents is available for analysis, in order to construct the entire feature space, and to compute statistics on the usage of certain words across the entire collection. For example, the TF/IDF measure requires all term frequencies in the collection to be known before constructing a feature vector for a document (or a query). The sheer size of the collection of Internet documents, as well as the dynamic nature of the collection, makes measures based on corpus–wide statistics impossible.

2.1.6 Search on the World Wide Web

The major search engines employed on the World Wide Web use traditional IR models based on full text indexing of documents encountered by their web robots, but commercial systems give few details about the specific models, distance measures, features, and relevance ranking methods they employ. Since the exact parameters of the model are unknown, it is hard for users to design optimal queries, even if they are information retrieval experts.

Since no librarian or publisher has specified relevant keywords for a given document, the current search engines use some form of full text indexing (with values of

feature vectors based on different factors, such as giving more weight to words occurring in titles or hyperlinks). Typically recall is good (there are plenty of documents returned on almost any query), but precision is poor, since the search is similar to a Unix "grep" command (simply a search for the query string occurring anywhere in the text).

Marchiori's work [37] has recognized some of these problems and sought to improve the relevance ranking algorithm by considering the hyperstructure of a document as well as the content. In that system, information content is a function of the text and other text which can be reached via a certain number of hyperlinks (i.e., a number of mouse clicks) from the original text. It uses the premise that connectivity is important for determining a page's worth.

2.2 Organization of Textual Information

Recall that browsing is an important activity for information discovery as well as search. In fact, one could argue that it is the browsing capability of the World Wide Web that has led to its huge popularity (after all, ftp and gopher have been around for a long time, but neither of them saw the tremendous growth that the web did). A meaningful organization of resources is essential for effective browsing. In this section we examine various document organization schemes arrived at by document clustering and document classification, as well as personal browsing agents.

2.2.1 Global collection representation

In order to facilitate browsing by concept, an information item's representation must necessarily be different from the word based model described above, since individual words in isolation do not in general denote concepts. Effective browsing requires a means for a user with general information needs to browse collections of text by examining several sources which are topically related. In a library, a user with a broad, general information need can go to the area where sources on the topic are kept, and briefly examine several to find text that meets the need.

Organization of entire collections of information conceptually requires a universal, global topic categorization. Examples of such organizations in the library context are the Dewey Decimal System [2] and Library of Congress Subject Headings [3]. Examples from the Internet/Web are directories such as Yahoo! [4] and Infoseek [5]. All of these global categorizations require two things that are difficult to automate: (1) an agreed upon categorization scheme and (2) assignment of textual sources to the appropriate category.

Global representation is intended to provide topical proximity of textual information, in order to facilitate browsing. It provides the mapping of document location to topic. To use a global representation to fulfill an information need on the Internet, you must know and understand the categorization scheme used.

2.2.2 Subject Catalogs

There have been some attempts to create subject catalogs for the World Wide Web. The most extensive and successful of these is Yahoo! [4]. Yahoo! gives a catalog of web pages based on a hierarchical topic structure. The catalog is constructed according to the standard librarian approach; a human must decide on the appropriate categories and assignments. Since it relies on human categorization, it's catalog is easy to browse if you have an idea of the topic you're looking for, but it suffers from an immense bottleneck of human evaluation. In a library, a document's content never changes, and there are relatively few new documents added in a given period of time. However, on the web, there are so many new documents added (and already so many documents in existence), that the human catalogers can not hope to keep up. In November 1996, Yahoo indexed 185,000 documents compared to AltaVista's 21,000,000 [38]. As of mid 1997, Yahoo! indexed 500,000+, but AltaVista had grown to 100 million [39]. Another drawback of catalogs is that there is no well–defined system of categorization, so it may be difficult for a user to determine in which category to look. For example, information retrieval stemming algorithms are found in Yahoo under library science and not under computer science.

Since topical hierarchies provide guidance in narrowing down a topic of interest, they result in higher precision than the full text query based search services, but since human effort is involved, there is a bottleneck to getting more information categorized. The human evaluation factor results in lower recall. Yahoo does not attempt to categorize the entire web, rather their goal is to provide high quality

organization for those sites that they do index.

There are also human constructed catalogs for small specialized domains (e.g., Biosciences [3], manufacturing [4]), which are sometimes subsets of the larger catalogs. These domain specific catalogs have the advantage that their catalogers have a smaller subset to work with, but there is still no unified standard categorization.

2.2.3 Document Classification

Document classification is the attempt to automatically place documents into pre-defined categories. These categories may be a globally imposed hierarchy such as Yahoo!, or merely the two categories of "relevant" and "not relevant" for a given user. Classification into global categories is of interest in the Information Retrieval and Machine Learning communities. Most methods require a large number of training samples and assume a fixed global document collection.

Letizia

Letizia [40] is a software agent that recommends web pages for a browsing user by concurrently exploring links from the page a user is currently browsing. It is actually performing dynamic web page classification into the 2 categories "interesting" and "not interesting". It does not use any query based approach for resource discovery. Rather, it follows links from the user's current page and recommends those that it believes the user will find interesting. It infers the user's interest based on observation

[3] http://golgi.harvard.edu/biopages/all.html
[4] http://www.nemonline.org

of the user's browsing behavior. Strong interest in a given page is indicated by saving a reference to it (i.e., "bookmarking" the page). Following a link from a page or returning to the same page several times also indicates interest. Not following a link, but following a link occurring later in the document can indicate disinterest in the link that was "passed over".

Letizia models the content of a document as a list of keywords. It provides a preference ordering of links to follow to assist the user. It does not require explicit relevance ranking of pages by the user, rather it infers relevance based on assumptions about the user's behavior.

Fab

Another system for providing page recommendations is Fab [41]. The Fab system utilizes both the content of pages and the recommendations of other users to determine its recommendations. Rather than recommending pages directly linkable from the user's current browsing position, Fab uses a set of independent software agents which collect information from the web and try to match a user's interest profile in determining which to recommend. The user must explicitly rank all pages suggested by the agent in order to provide feedback and also to provide the evaluation of the effectiveness of the system.

Fab uses the traditional information retrieval vector space model using a *term frequency - inverse document frequency* (TF/IDF) measure for the features. Recall that we described above some of the problems with adopting the information retrieval model for web-based information discovery: namely the nonexistence and impracti-

cality of measuring statistics from the entire collection, and the scarcity of labeled training samples relative to the entire collection. Fab avoids these problems by taking a random sample of 5,229 web pages, and considers that set to be the entire document collection for purposes of defining the global feature space. It then computes the maximum term frequency in a given document, and the document frequency for a given term. The 5229 web pages yield a fixed dictionary of 70,000 stemmed terms (after stop words are removed), weights are computed for each term using the TF/IDF, then the top 100 weighted terms are selected as features for future analysis.

Thus a 100-ary vector is used to represent both documents and the user profile (corresponding to a query in traditional information retrieval). The user profile is updated according to the formula $\boldsymbol{m} = \boldsymbol{m} + s\boldsymbol{w}$, where $\boldsymbol{m}$ is the user profile (a term vector), $\boldsymbol{w}$ is a document vector, and s is an integer $-3 \ldots 3$ representing the user's relevance score for that document. Several different agents are described, including search agents which explore the web [42], index agents which use existing web search services, and non–adaptive agents which consider "random" pages from various sources.

One of the problems with the Fab system is its reliance on the explicit human evaluation of each suggested page. It is likely that, especially in the early stages of learning, the user will be presented with many irrelevant or uninteresting pages, and the user will not want to bother ranking those, and may become frustrated with the high time investment with low payoff. Experience with Fab has also shown that the users are very strict in their rankings of pages – usually either very high or very low, rather than using the full range of the 7 point scale.

WebWatcher

WebWatcher [43] is a personal agent that attempts to learn which hyperlinks will lead to target information. Like Letizia, WebWatcher observes the user's current browsing behavior in order to infer relevance information about documents. However, unlike Letizia, WebWatcher assumes the user has a specific goal (in the form of a scientific paper, a specific domain) in mind, and observes a user's behavior to learn how to predict whether a hyperlink will be followed or not. Though the authors call this web search, they are not classifying documents. Instead, they are classifying hyperlinks into 2 classes: followed vs. not followed. They consider only information from the text of the current page, and not the content of the target of the hyperlink. It may not follow that these followed vs. unfollowed links correspond to relevant/irrelevant pages. WebWatcher uses a fixed length (530) boolean feature vector composed of 4 subvectors. The words comprising the features are determined during the learning phase. The subvectors are used to distinguish words of 4 categories: words occurring in hyperlinks (the anchor text), words occurring in the sentence of the hyperlinks (i.e., click here for information about machine learning), words in the headings associated with the hyperlinks, and words used to define the user goal. Feature selection for the first three subvectors was performed by ranking all the words in the training set according to their mutual information with respect to correctly classifying the training data, then choosing the top N words in the ranking. Mutual information is a feature selection technique where each feature is taken in isolation and tested on its ability to classify the training data.

Preliminary results were given from data gleaned from 30 browsing sessions of 3 different users who started from the same page, and had the same kind of goal (a specific kind of scientific paper). 23 of the 30 sessions resulted in a paper being found. Evaluation of the learning performance of the system was done by a comparison of the system's recommended hyperlinks with the links that the user actually followed. The best performer recommended as its first choice the link that the user actually chose 30% of the time, and the user's choice was in the classifier's top 3 choices 54% of the time.

One disadvantage to this approach is that it does not consider the content of a linked document, as mentioned above. Another is that it must have a specific target goal, and training is only relevant for that goal. For every new goal a user specifies, training (including feature selection) must start again from scratch.

Hotlists and Coldlists

Pazzani, Nguyen, and Mantik [13] describe an information filtering agent that tries to learn a user's interests based on their ranking of pages. In their experiments, a single user was asked to classify a set of URLs from a single domain (BioSciences) as interesting (the "hotlist") or not interesting (the "coldlist"). Out of 120 total URLs, the user selected 38 documents as positive examples (interesting) and the remaining 82 were negative examples (uninteresting). Sets of varying sizes, ranging from 10 to 100 were randomly selected to be training patterns, with the remaining patterns used as test data. Features were selected from the training data based on the expected information content of a word, with the idea that the most discriminating features

would be those that occur with very different frequency between the two classes. The 128 most informative features (words) were selected to form binary feature vectors, indicating occurrence or non-occurrence of the word in the document.

Error estimation was based on the classification accuracy of the test data. Four different learning algorithms were compared, a Bayesian classifier, a nearest neighbor algorithm using the matching coefficient as a distance measure, a nearest neighbor algorithm using a *value difference metric* [44] as a distance measure, and the ID3 decision tree algorithm.

Results showed that the Bayesian classifier clearly outperformed the other algorithms, achieving 85% accuracy with only 15 training examples, and 91% accuracy with 100 training examples. None of the other algorithms ever exceeded 78% accuracy. The authors conjecture that the superior performance of the Bayesian classifier may be due to the difference in importance of various features accorded by the conditional probability of each feature given the class, where the other algorithms treat all features equally, and the fact that the Bayesian classifier examines all features at the same time (in contrast to ID3). However, they do not mention the fact that the prior class probability is also considered for the Bayesian classifier, and the global pool of examples has $P(hot) = .317$, and $P(cold) = .683$. The decision rule for the Bayesian classifier is thus already predisposed to decide "uninteresting", and with no other information at all, it would be accurate nearly 70% of the time.

For the general information discovery problem, the prior class probabilities of interesting / not interesting can not be computed, but are likely to be even more skewed toward uninteresting than 30/70. If the prior probabilities were 5/95, one

could use a classifier that always decided "uninteresting" and report a 95% accuracy rate.

Note that the success of learning the profile may be due to the fact that all examples were from a single domain, which likely has a specific vocabulary. Note also that the assumed universe for feature selection was only the pool of interesting and uninteresting training data, and the features were chosen as those which were most discriminating across only this training set. While this technique may have produced accurate classification in a specific domain, it is questionable whether the features used for classification would be of any use in seeking additional information that was not present in the initial set of data.

2.2.4 Document Clustering

Document clustering also organizes documents into groups, but the groups are not according to some predetermined labeling; they are groups whose members are similar in some way, and dissimilar to members of other groups. Although there are not labeled categories (except for some attempts to label clusters), an organization of this kind enhances browsing once the user has found *something* of interest by providing topical proximity.

Visual Organization

Kohonen's WEBSOM group at Helsinki University of Technology use a self organizing map (SOM) to present on–line documents for easier browsing and exploration [45], [46], [47], [48]. A self–organizing map is a neural network technique in which high

dimensional statistical data is arranged in an easily visualized spatial map such that like inputs are near each other on the map. The first step is to create a SOM of *word categories* from the example documents using a local neighborhood of context (i.e., words that appear nearby) from the full text. The local neighborhood vectors are input into the neural network, which creates a map of word categories. Then individual documents are encoded as a histogram of the word categories, and a SOM of the encoded documents is created which can be browsed. This technique has been used to organize Grimm's fairy tales [45], articles from the single Usenet newsgroup "comp.ai.neural-nets" [48], and articles from 20 different Usenet newsgroups (which used a partially supervised method to increase the separability between groups) [47].

For the newsgroup "comp.ai.neural-nets", there were 4600 documents and 1700 words considered. The maps were created on specialized hardware, the massively parallel CNAPS neurocomputer, and then fine tuned with SOM software.

One problem mentioned in connection with organizing Usenet articles is the fact that they contain both short query or answer postings, which may not provide enough context to properly categorize them, and long articles which contain multiple topics, which properly belong in multiple locations.

HyPursuit

HyPursuit is a hierarchical network search engine described by Weiss, et. al [49] which gives an architecture for organizing the web which clusters hypertext documents in a "given information space" to aid browsing and searching. They propose a hierarchical organization where each information server's documents are clustered, and the

server keeps track of the summary information on it, so that queries can be routed to particular servers which are likely to contain the information. This scheme is in contrast to current existing search engines, which construct a massive single database of information on the entire web. One of the main aims of the HyPursuit project is to distribute the indexing information and then group the indices in order to alleviate network load and provide a coherent organization. Unfortunately, this proposed scheme requires universal agreement on the new paradigm, and sites which do not follow it will still require other methods of information discovery. The interesting aspect of the HyPursuit work is in its definitions of document and cluster similarity. They not only use document contents (the terms contained in the documents), but also hyperlink information. The idea is that if several documents $d_1, d_2, ...d_n$ all contain a link to another document d, then they are likely to be semantically related. Similarly, if several documents d_i are linked to *from* the same document d, the d_i are semantically related. Consequently, the definition for document similarity contains terms for both *content similarity* and hyperlink structure similarity. The idea that there is semantic information in the hyperlinks of a document is worth investigating further; however, it is difficult to discern the link structure for the entire web unless only small subsets are considered at a time.

Syntactic clustering

Rather than clustering documents based on semantic concepts, or topical categorization, Broder, Glassman, and Manasse clustered web documents based on their *syntactic* similarity [50], with the goal of finding documents that are "roughly" iden-

tical. For example, documents which occur in the same cluster may differ only in formatting, minor corrections, webmaster signature, or logo. Their goal is therefore not enhancing information discovery on the web, but applications such as updating widely used information (such as FAQs which occur on multiple sites), identifying intellectual property violations, or finding a document which has changed locations. We are more interested in semantic clustering, and will not further discuss syntactic clustering.

Topic Categorization

Martin [12] describes a system for clustering documents into overlapping hierarchies using the *factor learner* algorithm. The hope was to achieve groups of documents which were comparable to the Yahoo! categories. In this algorithm, each of several different groupings of n documents is called a *factor*. The factors are found by constructing an undirected graph consisting of all features, where a link between two features indicates a dependency relation, and no link between features indicates independence. The factors are then formed by finding cliques in the graph. Once the factors are found, the objects are clustered according to the factors using a greedy approximation of a Bayesian clustering algorithm. The method estimates highly probable conditional dependencies between factors and features.

Martin constructs a stop word list based on the most frequent words across the collection, augments keywords using WordNet [51], and uses frequency for feature values. He does not specify how many features were ultimately used, and describes preliminary results of clustering 54 documents from a single Yahoo category. Using

only words from the documents as features, the resulting clustered hierarchy matched the Yahoo hierarchy with only a few shared headings. Using the augmented words from WordNet, the resulting hierarchy shared half the headings with the Yahoo directory.

SONIA

Sahami, Yusufali, and Baldonado [52] describe the SONIA project, which clusters documents which were returned as the result of an initial keyword search. They use partitional clustering rather than hierarchical as in [12]. (Of course, partitional methods can always be applied iteratively to achieve a hierarchy of clusters.) Their goal is to aid search and browsing by dynamically organizing the results from an initial keyword search. They select features from the documents by first eliminating common English stop words, very infrequent words, and then words with minimal entropy. They use a Boolean representation for feature vectors (indicating presence or absence of a given word in the document). They then cluster the documents using both a Bayesian network (AutoClass) and a K-means algorithm. Once the clusters are found, they extract "descriptive" keywords for each cluster, based on the cluster centroid, or frequency of the words in each cluster. They present results from a query on "mining" which led to 50 documents and 3500 features, and a query on "Mars", which led to 39 documents and 1400 features. Results showed plausible clusters of subtopics within the query domain (i.e., mineral mining vs. data mining, pictures of Mars vs. life on Mars).

Scatter/Gather

The Scatter/Gather browsing technique from Xerox PARC [53] creates a hierarchical structure for navigating through large text collections. The original intent of Scatter/Gather was not to create a search tool to replace current information retrieval techniques, but to communicate information about the topic structure of a collection to a user. The interface for Scatter/Gather is that the user is presented with a set of clusters of documents with a summary of their contents based on topical words and typical titles of the documents they contain. The user can select one or more of the clusters which are of interest (*gather)*, and then ask to have these chosen documents re–clustered (*scatter*). Scatter/Gather clusters documents based on pairwise document similarity, and the cluster hierarchy is pre–computed off line in order to provide constant time access. Results indicated that Scatter/Gather is not superior to other methods of information retrieval when the goal is to locate specific documents. However, indications are that it does effectively communicate topical information structure to a user about a given document collection.

Further research with Scatter/Gather indicates that document clustering is an effective way to organize a document collection which assists the user in browsing a collection while avoiding the need to express their information needs as a formal query. More recent work has been done on clustering smaller collections, such as those returned from an initial search, and on efficient clustering algorithms so that the clustering can be performed in real time [54, 55, 56].

Personal Bookmark Clustering

Maarek and Shaul [57] have attempted to provide an organization of a user's bookmarks into conceptual hierarchical categories. This task is characterized by the fact that the concepts are personal and may be different for each user, the set of documents to be organized is a small subset of the entire document collection, and the collection is highly dynamic. They use a hybrid of manual and automatic organization where the user can control the behavior of the system, specifying which parts should be automatically grouped, and what should be left under the user's control. A user can specify a node in the cluster hierarchy as "frozen", which the system then considers as a black box sub–tree, and will not reorganize documents classified under that node.

An interesting aspect of this work is that instead of single words as features, it uses pairs of words which are linked by a *lexical affinity*, or correlation of their common appearance. They use a binary feature vector and do not specify the precise distance measure, but say it is a direct function of how many features the documents share. A hierarchical agglomerative clustering technique is used, and the problem of where to cut the dendogram is automatically determined by identifying the level clusterings within the tree that have a comparable degree of intra–cluster similarity. This was found to have more meaning for users than forcing them to specify a cluster size.

2.3 Summary

In this chapter we have described some of the current research into automatic information discovery from the World Wide Web. We described the models from traditional information retrieval, which are used by most of the current Web search services. We also described some of the assumptions about the library based information retrieval models, and characteristics of Web based information which require reconsideration of "standard" information retrieval search techniques.

Existing search services are intended to be applicable to the general user, and are not tailored specifically to an individual's preferences and interests. Browsing assistants consider the personal preferences of their users, but are limited in what they can recommend by either the pages directly accessible from the user's current position, or by pages which have been recommended by others.

We want to utilize services which cover the entire information space of the World Wide Web, and also personalize the search according the user's interests and preferences. By using a conceptual representation of information that is tailored to the current information need/interests, rather than relying on global conceptual ideas or isolated words as concepts, we will be able to perform automatic search and provide document organization which are based on concepts.

Chapter 3

Finding Concepts

In this chapter we describe the process used to define our intermediate level features which represent or approximate conceptual information. Remember that a user first provides some example documents that they determine are relevant or interesting in some way. These may be organized into categories (e.g., as a Netscape bookmarks file, or different email folders), or simply as an unlabeled list of files. The documents are fetched from remote servers if necessary and copied locally, where the *words* are parsed and counted. A word is defined as a sequence of letters without punctuation, digits, or white space.

Our next task is to find features that will define semantic concepts that are meaningful to the user from these example documents. Semantic concepts serve two specific purposes in aiding information discovery: they provide an organization or classification of documents, and provide a means for searching for additional similar information. We make an initial approximation of semantic concepts by finding important keywords from the example documents. Using pattern recognition techniques and

terminology, and consistent with the vector space model, we consider each document to be a pattern, with the words in all the documents as features.

Initially, we tried to find the important keywords by using feature selection and feature extraction of the words from the example documents, with the measure of success being accurate classification of the example documents into the user's categories. These efforts are described in detail in section 3.3.1 and section 3.3.2. However, our results showed that although these techniques may lead to accurate classification of the training data, the features (i.e., word subsets) found are not useful for searching, and do not provide any advantages in representation of documents. Subsequently, we used a different approach, described in section 3.4, that clustered the words according to their use across the document set, and then combined groups resulting from multiple clusterings into fuzzy sets. This technique creates groups of words which together suggest semantic content, and can be used as the keywords which approximate underlying "concepts" contained in the example documents. Results from word clustering on four different data sets are reported in chapter 4. By turning the focus away from document classification as an evaluation criteria for keyword determination, we can create sets of words which can be used both to seek out additional documents which were not in the training set, and to classify and organize topically the existing documents as well as new documents.

The next step in the process is to present these word groups to the user in order that they may further define the concept that is characterized by a given cluster. The user may tag specific words in a cluster as useless (unrelated to the topic), may ask for large clusters to be further subdivided, or tag more than one cluster to be merged

into the same topic. In addition, the user can provide a concept label to the cluster. This is not required, but can aid the user in further browsing of documents.

The user's participation at this point is not absolutely necessary; the system can use the topic definitions it found automatically without additional refinement, and can automatically create labels based on the individual words in the topic that have the strongest membership.

Once the user is satisfied with the concept word clusters, they form a basis for a new approach to various problems:

- automatic query construction for queries to existing search services
- new methods of indexing
- classification of query return results
- classification of previously unseen documents
- organization of existing collections according to personalized preferences

3.1 Preprocessing

We parse each document for *words*, defined as a contiguous string of letters, ignoring HTML tags, digits, and punctuation, as well as common English words (such as *the, of, and, etc*) from a pre–defined stop list. Our stop list is the one given by Fox [18], which is based on a broad range of literature in English.

3.1.1 Stemming

We can also reduce each remaining word to its word stem, so that words like *computer* and *computing* both become the same term, *comput*. This is a standard information retrieval technique, and we used the algorithm from Frakes [19]. If stemming is done, all unique word *stems* from the entire training set of documents form the global feature space. Without stemming, all unique words comprise the global feature space.

Stemming the words has the obvious advantage of reducing dimensionality of the feature space, for example, one of our data sets contained 7633 distinct non–stemmed words, and 5190 stemmed words. It would also seem advantageous in classification, since words with the same stems should be related to the same semantic concept, so we should count them as the same feature without regarding slight variations in tense and usage as distinct features.

However, there are cases where the simple stemming algorithm reduces words to the same stem when the originals were not related. For example, "animal" and "animation" are both reduced to the same stem, "anim", which merges two words which denote distinct underlying concepts into a single feature. A second disadvantage is that stemming is not a one-to-one function, so there is no inverse function from the stemmed word to the original. Once the original term is lost, it can not then be used in subsequent searching. A third disadvantage is that the best stemming algorithms are often inaccurate; they use heuristics about typical English word formation which do not always lead to correct results. For example, an "ed" ending on a word like *farmed* is removed to form the stem *farm*. But in the word "feed", the

"ed" is removed to get a stem of "fe", which is meaningless. Our initial experiments used stemming to reduce the initial feature space as much as possible, but we have subsequently found that the additional number of non-stemmed keywords do not significantly affect processing time nor results, and the utility of having complete words for generating automatic searches outweighs any advantage brought by the decrease in the number of features. Furthermore, experimental evidence shows that words which really are variants of the same root word (such as *america* and *american*, or *job* and *jobs*), appear together in the final word clusters, so their indication of similarity is preserved without resorting to stemming.

3.2 Forming the Pattern Matrix

The n example documents contain a total of d distinct words, so there are n patterns in d dimensional space, with $d >> n$. The $n \times d$ pattern matrix M is formed as follows: $M_{i,j} = k$, where word j appears k times in document i. One of our data sets, referred to as the "NEM-Online" data, consists of 85 documents in 4 categories containing 5190 distinct stemmed words. The documents are from the manufacturing domain, labeled by experts into the categories "labor", "legal", "government", and "design". This data set will be more fully described in section 4.1.

A portion of the NEM Online pattern matrix is given in Table 3.1, with rows representing documents (patterns), and columns representing word stems (features). Selected features out of the 5190 total feature space are shown for illustrative purposes. Rows which are not shown have 0 values in all positions of those columns.

Category	Doc Num	action	affirm	discrimin	drug	fda	banana	central
	25	7	0	0	0	0	0	0
	27	7	0	0	42	33	0	0
legal	28	53	0	0	2	5	1	0
	29	3	0	0	0	47	0	0
	30	1	0	0	5	27	0	1
	41	2	0	1	0	0	0	0
	42	1	0	10	0	0	0	0
	44	1	0	0	0	0	0	0
	45	2	0	0	0	0	0	0
labor	46	31	31	15	0	0	0	0
	47	52	50	12	0	0	0	0
	48	4	4	0	0	0	0	0
	49	2	2	0	0	0	0	0
	50	51	49	23	1	0	0	2
	53	5	0	0	0	0	0	0
	58	4	1	0	0	0	0	0
	59	5	0	0	0	0	0	2
	60	2	0	0	0	0	0	0
government	66	0	0	0	1	0	0	0
	68	2	0	0	0	0	0	0
	70	0	0	0	3	0	0	0
	71	2	0	0	4	1	0	0
	72	14	0	0	0	0	0	0
	76	0	0	0	0	0	0	1

Table 3.1: Sample of the partial pattern matrix.

3.3 Finding Important Features

If we attempted to classify these documents using their words as features with a parametric statistical classifier, the standard rule of thumb is that you need 10 times as many training samples as features for each class to accurately estimate the parameters[58]. Assume a user has collected 100 documents in 2 classes. Using a conservative estimate of 5000 distinct words in the 100 documents, a parametric statistical classifier would need 100,000 labeled training samples to estimate the

parameters. Clearly this is not possible with so many features.

Even though our ultimate goal is not merely classification of documents but detection of underlying concepts, a reasonable hypothesis is that features which successfully classify the user's training set will reflect the distinguishing characteristics of the document set and lead to the concepts. We therefore used some feature selection and extraction techniques to reduce the number of features, and used the classification accuracy of the training set as a measure of the worth of the features. The assumption is that a feature set which best reproduces the user's classification (has the highest classification accuracy) is the set which contains the "best" features. This assumption turned out to be false, and our experiments which led to this conclusion are detailed below.

3.3.1 Feature Selection

Feature selection is the process of choosing a subset of the available features for use in classification. There are 2^n possible subsets of n original features, so exhaustively enumerating all subsets is computationally prohibitive, especially when n is on the order of thousands. Some effective conventional methods for feature selection are sequential forward selection [59], sequential floating feature selection [60], and genetic algorithm search [61, 62]. We applied sequential forward selection and genetic algorithm search on our sample manufacturing data set.

Sequential Forward Selection

Sequential forward selection was applied using a 1 nearest neighbor classifier, Euclidean distance, and leave one out testing. The classifier performs as follows:

1. Select a pattern from the data set to be a *test* pattern whose category label we wish to assign.
2. Find the pattern vector from all the remaining data whose Euclidean distance is closest to the test pattern. This is the *nearest neighbor* pattern.
3. Assign the test pattern the category label of the nearest neighbor.
4. Repeat for every pattern in the data set.

The sequential forward selection algorithm first chooses the single best feature which gives highest accuracy in classifying the training set. It then successively chooses one feature at a time which, when combined with those features already chosen, gives the highest accuracy.

Using all 5190 features (stemmed words) in the four category NEM—Online data (described briefly in section 3.2) with the above classifier resulted in an error rate of 50/85, or accuracy of 41.18%. Using an SGI Indy[1], sequential forward selection achieved 17/85 errors, or 80% accuracy by selecting only 13 features. These features were *engin, action, david, contempl, affirm, architectur, ave, osha, abund, rehabilit, notic, commerc, transact.*

[1]Because the implementations of the C library routine quicksort behave differently in the presence of ties on an SGI and a Sparc Ultra, we achieved similar classification accuracy with a similarly sized feature set, but very different features. On a Sparc Ultra, sequential forward selection achieved 14/85 errors (83.5% accuracy) with 15 different features. Those features were *law, act, affirm, govern, drive, version, reduc, benefit, labor, bank, ai, omer, regul, edgar, label.*

Selecting any number of features sequentially, we were able to ultimately achieve 7/85 errors, for a 91.76% accuracy rate with only 17 features. Those features are *project, job, staff, david, apr, clinton, govern, gov, sophist, affirm, favor, commission, eeo, fiscal, cfo, tm, stori.*

Since we have such a sparse pattern matrix (zero values for many of the features in any given pattern), if we only add one feature at a time to the current pattern vector, there is a high probability that for a given test pattern, many other patterns will have a zero value for that feature, meaning that all of those patterns are equally far away from the test pattern. There will be many neighbors who are equally near the test pattern. This will be true for most distance measures and any sequential feature selection method. In other words, any sequential feature selection algorithm could produce many different subsets of features which are equivalent in classification accuracy by arbitrarily choosing different features in the presence of many ties.

Genetic Algorithm Feature Selection

A genetic algorithm is a technique for searching for optimal solutions to a problem based on the principles of natural selection [63, 64]. A solution to a particular problem is encoded in a bit string called a *chromosome.* Multiple solutions are initialized randomly in a single *population* and evaluated according to problem specific criteria. All chromosomes receive a fitness value according to the "goodness" of the solutions they represent. Three basic operations are applied to a population of solutions over several generations. They are selection, crossover, and mutation. Selection methods are used to ensure good solutions are more likely to survive to the next generation than

bad solutions. Crossover involves choosing two chromosomes in the current generation to use to construct a new chromosome in the next generation which contains some parts of each parent. Mutation is used to randomly change some small part of a solution (usually a single bit) in order to maintain population diversity and make sure the population does not converge to a single solution too fast.

Solutions which are more fit are more likely to survive and recombine with other good solutions in subsequent generations, while poor solutions tend to die out of the population.

Genetic algorithms have been used for the feature selection problem [61, 62]. The representation of a solution in a genetic algorithm for feature selection is typically a chromosome which is the same bit length as the number of features. A bit is assigned a zero value if the corresponding feature is not selected, and the value one if the feature is selected. This allows the genetic algorithm to potentially search the entire 2^n possible subsets. Using this technique with $n = 5190$ dimensions is computationally prohibitive, even though all 2^n possibilities are not examined. Instead, we chose a subset of fixed size k, which could contain exactly k features. Then there are $\binom{n}{k}$ possible subsets, which is a more tractable problem for the genetic algorithm (but still can not be exhaustively searched). We implemented this form of feature selection using the GALLOPS genetic algorithm software from Michigan State University [2]. We chose $k = 25$, and let the genetic algorithm search for the best set of 25 from the 5190 features. In 300 generations, the best subset of features was found in generation 168 and achieved 21/85 errors for 75.29% accuracy. The feature set was *bliznakov, volant,*

[2] http://isl.cps.msu.edu/GA/software/software-index.html

law, biographi, permiss, cwatx, alaska, treatment, sodium, move, evolut, version, avex, darnell, thoma, photographi, build, pany, hospit, law, lexi, plamen, briefli, export, cite. Recall that these are stemmed words.

3.3.2 Feature Extraction

Another pattern recognition technique we can use to try to obtain accurate classification is feature extraction. Rather than simply selecting from the existing features, feature extraction derives new features by transforming the original features in some way.

The Discriminant Karhunen–Loève Projection

One approach used in pattern recognition problems where the dimensionality is much greater than the number of training samples is the Discriminant Karhunen–Loève (DKL) projection, proposed by Swets and Weng for face recognition [65]. This is similar to Latent Semantic Indexing [21] which has been proposed in information retrieval, and which also constructs new orthogonal features which are linear combinations of the original features. It is called "latent" semantic indexing because it claims to capture "latent" or "hidden" semantics of a document.

The DKL projection consists of first performing a principal component projection followed by discriminant analysis. The principal component projection selects as the first feature the linear combination of dimensions which has the greatest amount of variance among all the patterns. Then a new dimension is chosen which has the highest variance of patterns among all dimensions orthogonal to the first. So

we can project the original space into one whose dimensions are the eigenvectors corresponding to the largest eigenvalues of the original features which still retain some high percentage of the total variance.

With our 5190 stemmed features, only 23 eigenvectors were required to retain 90% of the total variance. Thus we transformed the 5190 dimensional original feature vectors to new 23 dimensional vectors. The 23 new dimensions are orthogonal linear combinations of the original 5190. Classification using these 23 dimensional feature vectors did not achieve very high accuracy, only 54%. This result is not surprising since the principal component method is only concerned with finding orthogonal axes in the feature space of maximal variance, and is not concerned with class separation.

However, the Karhunen–Loève projection is only the first step in the DKL projection. The 23 new feature vectors are next projected into another space using discriminant analysis, which does consider class separation of the training patterns.

Discriminant analysis considers each class of patterns in the n dimensional space, and tries to project the data into a smaller dimensional space while maximizing the between–class scatter and holding the within–class scatter constant [66].

The scatter matrix for a given class k, $S^{(k)}$ is defined as follows:

$$S^{(k)} = \sum_{j=i}^{n_k} (x_j - m^k)(x_j - m^k)^T$$

where x_j is a pattern vector from class k, m^k is the mean feature vector in class k, and n_k is the number of patterns in class k.

The within–group scatter matrix is the sum of the group scatter matrices:

$$S_w = \sum_{k=1}^{K} S^{(k)}$$

where K is the total number of classes, 4 in our case.

The between-group scatter matrix, S_b is the scatter matrix for the group means:

$$S_b = \sum_{k=1}^{K} \sum_{j=1}^{n_k} (m^k - m)(m^k - m)^T$$

To project the data into a t dimensional space, we retain the eigenvectors corresponding to the t largest eigenvalues of $S_w^{-1} S_b$. Note that in order for S_w to be invertible, we must have the number of patterns, n, greater than the dimensionality of the patterns, d. This is why discriminant analysis by itself will not work on the 85 patterns in the original 5190 dimensional space. So we use the 23 new features obtained by the Karhunen–Loève projection. We also note that there will be at most $c - 1$ non-zero eigenvalues in $S_w^{-1} S_b$, where c is the number of classes. So the final dimensionality of the space must be less than the number of classes.

Now in our data set we have $n = 85$, $d = 23$, and $c = 4$. We use discriminant analysis to project the patterns of the Karhunen–Loève projection into a 3-dimensional space.

With our data set and performing KNN classification on the combined DKL projection, we got 43/85 errors, or 49.41% accuracy with $K = 1$. This is a lower accuracy than even the Karhunen–Loève projection alone. With $K = 5$, we did only slightly

better, with 36/85 errors for 57.65% accuracy. Although the DKL projection performed well in the face recognition problem [65], it appears that the DKL projection is not robust enough to create a general classifier for document classification. For the DKL projection to be successful, one must train over all possible expected variations of the eventual data to be classified. The DKL based retrieval fails if the search probe varies significantly from any of the training examples [65]. Also, in the face recognition problem there were many more classes than 4. Since the final dimension is constrained to be less than the number of classes, the DKL projection may not perform well where the number of classes is so few.

Even if the DKL projection had adequately classified the training data, it suffers the drawback that its low dimensional features are created by linear combinations of the original 5190, which do not by themselves give any suggestion of semantic content. We can not therefore use eigenvector based features for subsequent search, since documents are not indexed in search engine databases based on those eigenvector features. Even if one could construct such an index for every document on the web, a new index would have to be constructed for every additional or new training example. The eigenvectors derived from the original training set will not generalize to other documents in other categories.

There is no way to examine only a subset of documents on the Web according to their weightings of 5190 features that happened to be in the training set. This is a very inefficient way to discover new information, especially since the vast majority of documents on the web properly belong to the reject class for a given information need. We need the ability to seek out documents which match a user's preferences

without examining the entire global set of documents, preferably using the resources that already exist, namely, keyword based indexes of documents.

Discussion

The feature extraction technique did not achieve high classification accuracy, nor did it suggest semantic content. Though the feature selection techniques achieved high accuracy in classification of the training data, the features in each subset were quite different and do not by themselves suggest the content of the original (or any) categorization.

It is not meaningful to select a feature as a category discriminator when there are many potential candidate neighbors for a test pattern with few features. Why then do the selected sets of features above achieve such high accuracy rates? The answer lies in the sparse nature of the original pattern space. With 85 points in 5190 dimensional space, there will be many piecewise linear decision boundaries which adequately partition the training patterns. However, the real question is whether these decision boundaries are meaningful or generalizable. We want not only to find some decision boundary, but the best general decision boundary, and meaningful features which will suggest the semantic content of documents, and enable us to find new data. We conclude that our assumption that the feature set which best reproduces the example classification is the best feature set for defining concepts was invalid. One of the problems with this approach is that individual words are used in isolation to represent document content and a globally imposed categorization label is used as the performance criteria. We are simultaneously using two extremes of

representation which do not have a good mapping to each other; we would prefer a unified conceptual level of representation between the two extremes.

3.4 Transposing the Matrix: Clustering Words Instead of Documents

Our efforts at classification accuracy using feature selection were somewhat successful, but the resulting features were not useful in seeking new documents or in suggesting semantic concepts. The problem lies in the low level representation of information at an individual isolated word level. We now focus on ways to find features which will aid us in finding new data, which will represent the documents at a higher conceptual level than individual words.

Since we know that the words in the original pattern matrix are correlated, we should be able to cluster the words into groups whose members are similar to each other and not similar to members of other groups. We may then be able to use the groups of similar words to form *concepts* which are meaningful representations of the information contained in the documents, and which can subsequently be used to aid search, and provide personal organization of documents.

Recall that in the $n \times d$ pattern matrix M, (Table 3.1), rows represent documents, columns represent individual features (words), and the feature values are the number of occurrences of that word in the given document. When we were trying to reproduce the example documents' classification, we were grouping documents according to their

pattern of words. Now, instead, we group the words according to their usage pattern across the sample document collection.

To group the words, we used a K-means partitional clustering algorithm [67], where K points are chosen randomly as the means of K groups in the n dimensional space. Each of the d points is then assigned to the group whose mean is closest. After each point is assigned, the group means are recomputed, and the process repeats until either there is no change, or after a fixed number of iterations. The K-means process is not guaranteed to find a global optimum grouping, so we run it several times with different random seeds, resulting in groups which often overlap. Combining results of these different groups to form fuzzy sets is described in section 3.4.4.

3.4.1 Choosing a value for K

One of the important parameters of the K-means algorithm is the choice of a value for K, the number of groups. If K is too small, there will be few groups which have many members. If K is too large, there may be empty clusters, or many small groups which should be grouped together. If some of the K clusters are empty, we can simply accept less than K groups. This means that the number of final groups, as well as their final membership, will depend on the initial random seed. Since we know the number of categories, c in the training set, we allow for two groups of words (i.e., 2 "concepts") for each document category, plus 2 groups for a reject category. Therefore we choose $K = 2(c+1)$. Experiments have shown that greater values of K tend to generate many empty clusters.

3.4.2 Distance Measures

Choice of a distance measure between patterns is also an important parameter in the clustering algorithm. We have used three different distance measures in our experiments, Euclidean distance, cosine distance, and Jaccard distance.

Euclidean distance

For two d dimensional vectors $X = (x_1, x_2, \ldots, x_d)$ and $Y = (y_1, y_2, \ldots, y_d)$, Euclidean distance is defined as

$$\left[\sum_{i=1}^{d}(x_i - y_i)^2\right]^{1/2}$$

and is the typical straight line distance between end points of the vectors X and Y in Euclidean space.

Cosine distance

If we again consider the points X and Y as vectors from the origin, instead of measuring the distance between the endpoints, we can measure the cosine of the angle between the vectors. Since in our case both X and Y will always have positive values, we are only dealing with the first quadrant of the space. The cosine distance is defined as:

$$\frac{\sum_{i=1}^{d} x_i y_i}{\left(\sum_{i=1}^{d} x_i^2\right)^{1/2}\left(\sum_{i=1}^{d} y_i^2\right)^{1/2}}$$

and is commonly used in information retrieval. The cosine distance has an appeal as a distance measure between *documents* where word occurrences are used as features, because points along the same vector will have the same distance to another vector, regardless of the vector lengths. This means that the proportion of words with respect to other words (the other dimensions in the space) is constant along a given vector, despite the *number* of words being more or less. This gives an automatic normalization for document length. Consider measuring *document* distance in a space with only two dimensions (i.e., two words), *a* and *b*. Suppose document d1 contains 2 *a*s and 2 *b*s, document d2 contains 2 *a*s and 4 *b*s, document d3 contains 50 *a*s and 50 *b*s, and document d4 contains 25 *a*s and 50 *b*s, as shown in Figure 3.1.

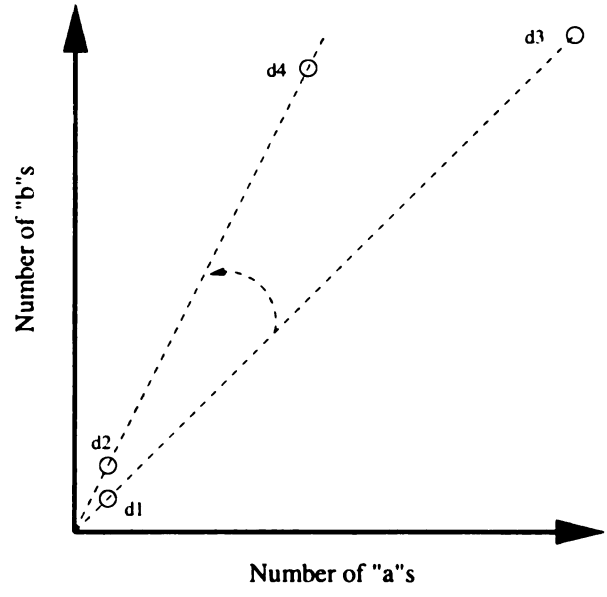

Figure 3.1: Four documents in 2 dimensions

Using Euclidean distance, points d1 and d2 are much closer to each other than d1 and d3, but there is 0 cosine distance between d1 and d3. The cosine distance from d1 to d2 is identical to the cosine distance between d1 and d4. However, when

measuring distance between *words* rather than documents. it does not make sense to consider a word w1 that occurs once in document d1 and once in document d2 to be identical to a word w2 that occurs 100 times in document d1 and 100 times in document d2 simply because their ratios are the same. We also see empirically that the cosine distance does not make sense between *words*, since our K–means clustering algorithm always failed to converge within 100 iterations when clustering words using the cosine distance.

Jaccard distance

The third distance measure we used was Jaccard's coefficient [66], using the binary values 1 for occurrence of a word in a document and 0 for non–occurrence. Jaccard's coefficient as a measure of distance between two *documents* is defined as

$$\frac{m_1}{d - m_0}$$

where m_1 is the number of distinct words that appear in both documents, d is the total number of unique words in the whole collection (the number of dimensions), and m_0 is the number of words that appear in neither document. This distance measure considers only presence or absence of a word, and ignores the frequency of occurrence of a word in a document. When using Jaccard's coefficient as a measure of distance between *words*, m_1 represents the number of documents in which both words occur, d is the number of documents, and m_0 is the number of documents in which neither word appears. Since the Jaccard coefficient is ignoring some potentially important

information, namely, the frequency of usage of a word in a document, we expect that the Jaccard measure will not perform as well as Euclidean distance. Again, empirical evidence confirms this, where we see that words fail to converge into clusters within 100 iterations when using the Jaccard coefficient.

3.4.3 Normalization for document length

As we mentioned, one attractive feature of the cosine measure (but only useful when measuring distance between *documents*) is that it automatically normalizes for a document's length, on the assumption that longer documents should not contribute more than shorter documents. When clustering words, we also wondered if we should normalize for document length, by dividing each word count in a document by the total number of words in the document (giving a frequency between 0 and 1 as a value for a given word rather than an integer number of occurrences). In other words, in our original pattern matrix, $M_{i,j} = k$ where word j appears k times in document i, we can normalize M by document length by defining M', where

$$M'_{i,j} = \frac{k}{\sum_{j=1}^{d} M_{i,j}}$$

Under the original scheme, longer documents weigh more heavily into the clustering than shorter ones. The K–means clustering was performed on the NEM-Online data which had been row normalized. Resulting clusters tended to be more polarized, with small clusters smaller and large clusters larger, and the semantic content was

less striking than when using word occurrences.

Normalization seems like a good idea but upon reflection, we realize that it is beneficial only if the documents are fairly homogeneous in their style. We know that online documents vary widely, from full text to a simple list of links. We conjecture that we need some length to the documents in order to extract context which will lead to grouping of words from the same concept. Long documents are more likely to reflect "typical" usage patterns than simple lists or abbreviated documents such as abstracts. Long documents are providing more information, so it is reasonable that they contribute more heavily to the clustering.

3.4.4 Combining runs into fuzzy sets

A mathematical set X with members x_i in a universe U can be defined as a function $f : U \rightarrow \{0, 1\}$ where $f(u) = 0$ if and only if $u \in X$ and $f(u) = 1$ if and only if $u \notin X$. In other words, set membership can be thought of as a function from all objects in the universe to the set {0, 1} where a 1 indicates membership and 0 indicates non–membership.

Fuzzy set theory was first proposed by Zadeh [68] as an alternative to sets with absolute memberships. Now, instead of mapping the function from U to the discrete set {0, 1}, we map U to the interval $[0, 1]$. Members of U can have varying degrees of membership, and f is called the *membership function.* If an object has a membership value close to 1, it is a strong member of the set, while a value close to 0 indicates a weak member. Note that this is not the same as the probability of a member being in

a set (though both have values in $[0, 1]$); if an object x_i has probability .5 of being in a set A and a probability of .5 of being in a set B, it still ultimately has a membership value of 1 in either A or B, but not both. In fuzzy set theory, it is possible for an object to simultaneously belong to multiple sets, and all membership values for all sets need not sum to 1.

Fuzzy set theory has been used in pattern recognition [69] as a means to arrive at clusters which have a fuzzy membership. Rather than forming fuzzy clusters at the outset, we find partitional clusters, then assign a fuzzy membership function to their objects, as described below.

We ran the K–means word clustering algorithm many times with different initial random seeds to increase our confidence in the results. Examining different runs by hand reveals that there are consistencies, though differences do exist. We need to precisely quantify the cluster set membership, and combine the results from differently seeded runs. Philosophically, concepts are not composed of a fixed set of words, neither are they stochastic. The underlying notion of a concept is fuzzy, so we use fuzzy set theory to define the final membership of a concept cluster [69].

To form "concept" clusters, we combine the results of several runs of the K–means clustering, in order to get a definitive assignment of words to clusters. Recall that we run the original K–means algorithm n times, with n different random seeds. Words that appear together in the same cluster all n times have strong evidence that they truly belong in the same cluster. Furthermore, words that co-occur in the same cluster in only a few runs probably do not ultimately belong together. We can use the number of times words co-occur as a measure of their strength in relation to each

other; co-occurring a large number of times shows that the words are bound together strongly, and are likely to be from the same "concept".

Since words have different senses, some words may validly belong in more than one concept. For example, the word "circuit" may occur in half the runs along with "design", "CAD", "electrical", and in the remaining runs may occur along with "court" and "appeals". The fuzzy membership function allows the word to occur in both concept clusters.

Our experiments have shown that there are usually $K-1$ "smaller" clusters, and 1 very large cluster that contains > 80% of the original feature words. This large cluster contains the very low frequency words which are not useful in discriminating between documents, so ignoring this cluster reduces the number of words without sacrificing discriminating power. To combine clusters from several K–means runs, we first find all unique words from the $K-1$ smallest clusters across all runs. We want to form "master clusters" which combine results of all runs, and which are actually fuzzy sets, with words appearing togther many times across several runs having a high degree of membership, and words appearing few times together with the other words in the set having a low degree of membership.

We have used two different methods to construct the fuzzy sets. The first method is as follows. Assume we already have $K-1$ master clusters, $M_1, M_2, \ldots M_{K-1}$. Given a new cluster C, we can find the master cluster with the maximum intersection, that is, find the M_i such that $|M_i \cap C|$ is maximized. Then M_i is updated by adding C to it, so $M_i = M_i \cup C$. However, we want to remember the intersection $M_i \cap C$, because the number of intersections will determine the fuzzy set membership. In practice, we

can keep track of the number of intersections merely by not eliminating the duplicate members when taking the union.

The fuzzy membership function for a word w belonging to master cluster M_i is defined as follows: $f_{M_i}(w) = dup/n$, where dup is the number of duplicates of the word within that cluster (i.e., the number of times it intersected with another similar cluster), and n is the number of runs we are combining. Thus all elements have a membership value between 0 and 1.

This approach is convenient and fast, but assumes the existance of initial master clusters. One way to obtain initial master clusters, is to simply accept the first set of clusters generated as the "master" clusters, then find the maximal intersection of subsequent runs with these initial clusters. This is a cheap and easy way to form the fuzzy sets, but has the drawback of being dependent on the order in which the runs are presented. If the initial run has anomalies due to poorly chosen initial centers, it will affect the final cluster membership.

To avoid this problem, our second method constructs a full co-occurrence matrix R for words in the $K - 1$ smallest clusters and forms the fuzzy clusters by finding the words with the maximum co-occurrence. All pairs of words in a cluster in a given run are given a vote each time they appear together in a run. So for n runs, the maximum value in the co-occurrence matrix $R_{i,j}$ is n, meaning that word i and word j appeared together all n times. The minimum value of $R_{i,j}$ is 0, meaning that word i and word j never appeared together.

Once the matrix R is constructed, the algorithm for forming fuzzy clusters is as follows:

1. Level = n

2. For all words i

   (a) Find all words j such that $R_{i,j}$ = Level

   (b) If i and j are not already in the same cluster, create a new cluster that contains words i and j.

   (c) Compute the set membership function *strength*, of i and j, for each new cluster they appear in.

3. Decrement Level by 1.

4. Go to Step 2.

The *strength* function of a word w in cluster C can be defined in a number of ways. One method is to use the average connectedness of word w to all other words currently in the cluster:

$$strength(w) = \frac{\sum_{i \in C} strength(i)}{|C|}$$

When any new word is added to a cluster, the strength of every word in that cluster must be recomputed.

Unfortunately, this algorithm is in the worst case $O(j * n^4)$ in time, and $O(4n^2)$ in space, where j is the number of runs and n is the number of unique words in the $K - 1$ clusters across all runs. We could make the algorithm faster by making a simpler strength function (such as $strength(w)$ = Level at which it joined the cluster)

but can in no case do better than the $O(j * n^3)$ of Warshall's algorithm required to find the transitive closure of the matrix j times. Ten runs for $n = 5000$ requires over 4 hours clock time and 100 MB of space on an Enterprise Server 3000 with 250 MHz Sparc Ultra 2 processors and 512 MB of main memory. This is a powerful machine, not readily accessible to an average user. Recall that one of our requirements is fast turnaround and easy accessibility. The use of this algorithm may be too resource intensive for average users.

When we have the fuzzy clusters constructed by either method, we can identify the "core concepts" in each by choosing the words with the strongest membership. These core concepts can now be presented to the user, who can identify whether key terms actually should belong to the concept, and who can modify the degree of fuzzy membership if desired.

3.5 Summary

In this chapter we have described methods for finding concepts in text. We have concluded that classification accuracy of a small set of training examples is not a good indicator of features that convey semantic meaning. Word clustering using several different runs of the K–means algorithm and Euclidean distance provides small groups of words which lead to an approximation of semantic content. In the next chapter we will discuss in detail the results of this process on four separate data sets, then describe the uses of the concept clusters for automatic search and document organization.

Chapter 4

Experiments and Results

In this chapter we describe our experiments with concept extraction and the results obtained using four different datasets. First we describe the data sets used, then the experiments which obtained the fuzzy concept clusters. We describe the convergence behavior, size of resulting clusters, and strengths of resulting clusters. We conclude the chapter with a discussion of the uses of the derived clusters in automatic search and document organization.

4.1 Sample Data Sets

Statistically Random

First, in order to provide a baseline for comparison and to ensure that we don't derive conceptual results from statistically random documents, we need a set of documents that are statistically the same as typical documents, but which are not conceptually related. We generated a set of random documents in the following way. We obtained

the Reuters-21578, Distribution 1.0 [1] test collection of documents. This collection consists of 21,578 Reuters newswire articles from 1987 which all deal with economic topics. There are 45,534 distinct words in the entire collection. To generate our purely random pages, we computed the frequency distribution of all words in all 21,578 articles, then sampled words with replacement according to their observed probability estimate. The word with highest probability, was, not surprisingly, "the". The overall probability of a word w appearing in a document is defined as

$$P(w) = \frac{n}{N}$$

where n is the number of occurrences of w in the entire collection and N is the total number of words in the collection. We created 80 documents with 1000 words in each document. The number of unique words in these 80 generated documents was 9238, with 4881 of those words occurring only once, and 1394 occurring only twice. The majority of all words are low frequency ones. Note that word order in the generated documents is not important, since we do not use phrases or word pairs in our analysis.

Seeded Random

Once we had the purely random documents, we "seeded" them with a set of chosen "topic" words in a controlled way, in order to make sure we could reliably retrieve these topic words using our methods. Each Reuters document contains zero or more category labels which were assigned manually by experts. A given article may contain

[1] available from David Lewis at http://www.research.att.com/lewis

many labels, and many of the categories could have overlapping concepts, or be conceptually hierarchical. For example, "corn", "wheat", and "grain" are all distinct categories in the Reuters data but many articles contain concepts from all three. Due to human decisions about labeling, it is possible for an article about wheat to be labeled "wheat" but not "grain", even though "wheat" can be considered a sub category of "grain". Though the Reuters documents have labels manually assigned by experts, there are many errors and inconsistencies in the labeling of articles. For example, one article about soybeans (without mention of meal or oil) may carry all of the labels "grain", "soy-meal", "soy-oil", and "soybean", while another similar article may have only the single label "soybean". Many articles from the "gas" category do not also have the label "fuel", and some articles are clearly mislabeled (one example is an article about the price of gold carrying the label "gas").

The Reuters data is also full of many abbreviations standard to Reuters, but not to typical English text, such as "dlrs" for *dollars*, "mln" for *million*, "pct" for *percent*, "bpd" for *barrels per day*. Additionally, there are many proper names and misspellings, and special routing codes for the news wire. All of these factors contribute to the phenomenon of a higher number of distinct words and a different word distribution than one would typically expect from, say, English literature.

The four Reuters categories *acquisitions, gas, interest, grain* were chosen, and the distribution of words in each category alone was computed to get the probability that a given word appears in an article of that category. The conditional probability of a

word w appearing in a certain class c is

$$P(w|c) = \frac{n_c}{N_c}$$

where n_c is the number of occurrences of the word w in category c, and N_c is the total number of words in category c.

When $P(w|c) >> P(w)$, the word should be an important feature of class c. We therefore took the words with the largest values of $P(w|c) - P(w)$ as "seed" words to insert in the purely random documents. For example, $P(oil) = .00158$, and $P(oil|crude) = .0177$. The difference indicates that *oil* is probably an important word for the category *crude*. The words with the greatest difference between overall probability and class conditional probability form our seed word sets for each category. It is interesting to note that the set of words with greatest differences between overall and class conditional probability often included words such as "the", "from", "a", which also have a very high overall probability. In the same way that we removed stop words from documents prior to word clustering, we removed stop words (from the same list) from the potential seed word sets. The top 10 "seed" words from each category is listed in Table 4.1. Note that even these small sets of words overlap categories, "pct" appears as a top class conditional word in both the categories "gas" and "interest".

There are many parameters we can control to construct seeded random documents, including document size, number of documents in a target category, seed word set size, percentage of the random document that is replaced with seed words, and the

acquisitions		gas		interest		grain	
lt	shares	pct	gasoline	rates	rate	tonnes	wheat
company	include	mln	oil	pct	bank	grain	corn
offer	corp	prices	bpd	market	money	agriculture	usda
stake	share	barrels	crude	stg	fed	nil	crop
dlrs	acquisition	demand	octane	cut	banks	department	export

Table 4.1: Seed words from four Reuters categories

percentage of the seed word set included in a given document.

To construct our random seeded documents, we constructed three sets of 80 documents of 1000 words each, in the four target categories mentioned above (20 documents per category). These values roughly correspond to our "real life" data sets, and so represent a realistically sized set from which to derive concept clusters. In set one, 10 words out of the 1000 (1%) were chosen uniformly from a size 10 seed word set. The remaining 990 words were chosen according to the overall collection distribution of words, giving a total of 8881 unique words. The second random seeded data set had 5% seed words (again, chosen uniformly from a seed set of size 10) with all other parameters the same, resulting in 8557 unique words among the 80,000. The third data set was 10% seed words (100 words out of 1000 chosen uniformly from a size 10 seed set), with 8346 unique words. We wanted to ensure that we obtain the seed words as concept clusters from each of the four categories.

Reuters articles

We next chose a subset of actual Reuters articles, chosen from the same four specific categories. Again, we wanted to use a small subset of example articles, to realisti-

cally simulate real life situations where a user in general does not have access to an entire document collection, and therefore can not compute the global probabilities or the class conditional probabilities. We therefore sampled *articles* from the chosen categories at random without regard to any probabilities, or any requirements of exclusiveness of the category label. In other words, we were not choosing the "best" representatives of a category which might be the case when an article has *only* that label. An article had an equal chance for selection as long as it contained the correct category label (from perhaps many labels). For each of the four categories *acquisitions, gas, interest, grain*, we randomly selected 20 documents for a total of 80 documents. The Reuters data is summarized in Table 4.2.

Category	Number of Documents	Distinct Non-Stemmed Words	Document Length			
			Min	Max	Avg	Median
acquisitions	20	1087	30	369	134	82
gas	20	911	27	355	107	96
interest rates	20	1087	27	388	134	92
grain	20	760	33	234	87	78
total	80	2665	27	388	115	110

Table 4.2: Sample Reuters data set categories

NEM-Online

The final data set we examined comes from the manufacturing domain, and consists of actual web documents from the Network for Excellence in Manufacturing (NEM Online)[2], which contains a set of documents manually selected by human browsers to be of interest to a particular user community. We again chose four categories,

[2]http://www.nemonline.org

labor, legal, government, and design, and collected 85 documents which we were able to retrieve from the Web. The Nem-Online data is summarized in Table 4.3. This data set is the most realistic, with varying numbers of documents per category, and varying lengths of the documents.

Category	Number of Documents	Distinct Non-Stemmed Words	Document Length			
			Min	Max	Avg	Median
design	23	2155	37	990	253	127
government	15	2852	16	5450	633	80
labor	13	2444	33	2557	552	342
legal	34	3921	34	3137	403	123
total	85	7633	16	5450	456	127

Table 4.3: Sample NEM Online data set categories

In the tables, document length refers to the total number of words in the document (excluding stop words). Notice that the Reuters articles are much shorter than the NEM Online documents. This makes concept extraction from the Reuters data more difficult, since there is not as much context in a given article to distinguish the characterizing concepts.

The category labels were assigned by human experts. Note that the categories may contain underlying concepts which overlap. For example, a document discussing "affirmative action" may be reasonably classified into government, labor, or legal categories, according to the person who is doing the category assignment, and within the context of their other collected documents. The NEM Online category labels are distinct, but recall that the Reuters documents may (and usually do) have multiple category labels. The overlapping of concepts into multiple categories is one thing

that makes text classification so difficult: even human categorizers do not have a well-defined, unequivocal method of assigning a definitive label to a certain document.

All sets of training data are summarized in Table 4.4.

Data Set	categories	docs in each category	total distinct words	conceptually related words
StR	1	80	9238	none
SeR 1%	4	20	8881	10
SeR 5%	4	20	8557	10
SeR 10%	4	20	8346	10
RA	4	20	2665	unknown
NO	4	see Table 4.3	2665	unknown

Table 4.4: Summary of the training data sets: Statistically Random (StR), Seeded Random (SeR) at 1%, 5%, 10% seed words, Reuters articles (RA), and NEM–Online (NO).

4.2 Word Clustering Results

In this section we discuss the results of clustering the words of the four different data sets, and convergence, cluster size, and word strength in the fuzzy clusters in each data set.

4.2.1 Statistically Random Data Set

Random data with no inherent clusters are manifested by patterns whose features are all similar. This is the case with the statistically random data. All documents look alike across the feature set, so we do not expect to get semantically related words as clusters. The K–means algorithm fails to converge after 100 iterations,

indicating that it is having difficulty finding stable clusters. When forced to stop after 100 iterations, we assume that the words within the clusters have little chance of being semantically related to each other. They represent those words whose overall probability of getting into a document is similar. Interestingly, the sizes of the clusters follow the same pattern as the non-random data do; we get one very large cluster and $K-1$ smaller ones. This is a result of the tendency of English text to follow a distribution according to Zipf's law [70], which states that the frequency of word use times its rank order in a given text is approximately constant [15]. Sizes of the $K-1$ smallest clusters of all data sets are shown in Figure 4.1.

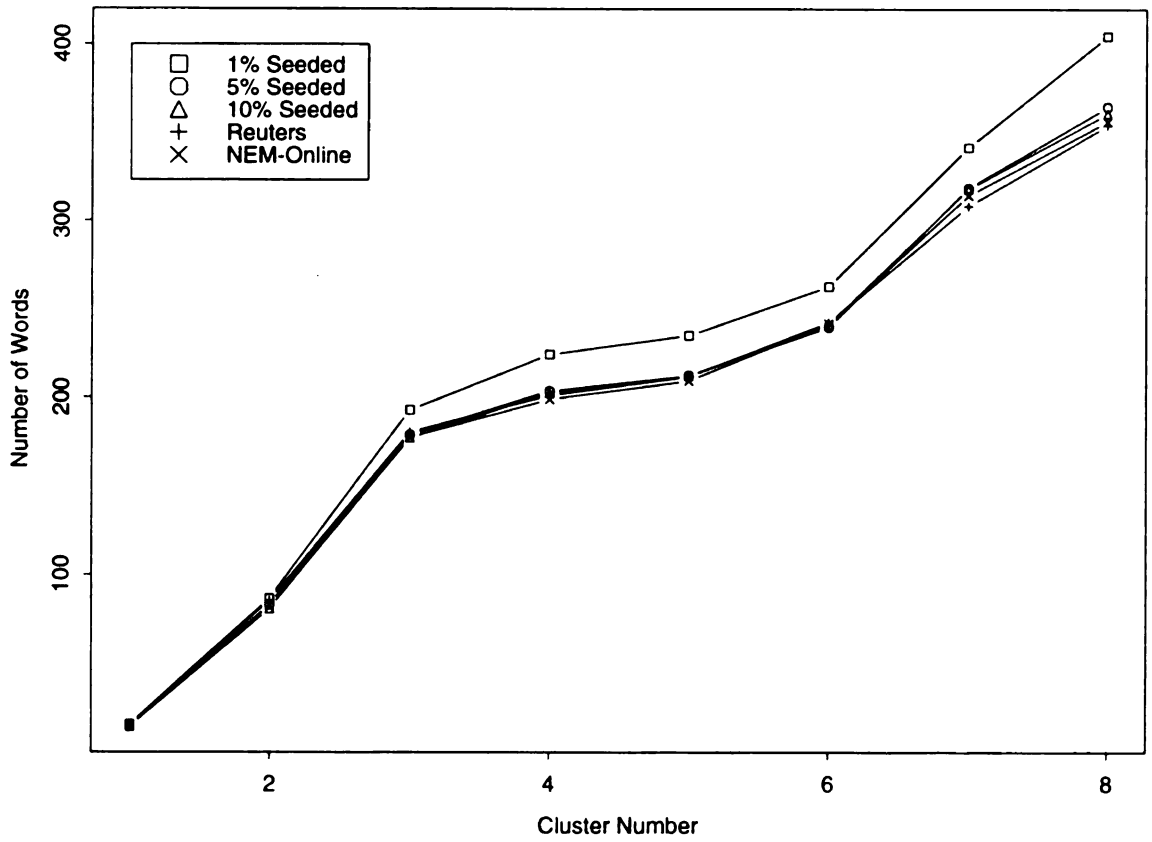

Figure 4.1: Sizes of $K-1$ smallest clusters in all data sets

With $K = 10$, the sizes of the clusters from the random data are roughly the same over all 10 runs, even though the algorithm did not converge. Furthermore, the 3 smallest clusters have sizes 9, 10, and 30 (or 31) in all runs. If we examine the contents of those smallest clusters, we see the same words appearing. The smallest cluster contains *bc, dlrs, lt, march, mln, pct, reute, reuter, vs.* Next smallest contains *april, bank, billion, company, corp, cts, inc, mar, net, usa.* If we examine the overall probabilities for these words in the Reuters collection, we see that the rank order of the probabilities for the first cluster is 15, 11, 12, 16, 14, 19, 30, 20, 26. The rank order for the second cluster is 46, 47, 36, 43, 41, 49, 45, 35, 50, 33. We are obtaining as clusters those words which have a similar overall probability, which is what we would expect. The words with highest overall probability (ranks 1-10) are all stop words: *the, to, of, f, in, and, a, said, s, for*, and thus were eliminated before clustering.

So, for the purely random data, we obtain clusters that contain words with similar overall probabilities, but which do not suggest any semantic content.

4.2.2 Random Seeded Data Set

Since the purely random data had no class information, and its documents were created based on the overall probability distribution of words, the clusters obtained were based on those overall probabilities. For the random seeded data, since we have specific words in four categories, with those words being the highest difference in class conditional probabilities, obtaining those seed words in our clusters will reflect the fact that the documents are differentiated by those words with the highest class conditional

probability differences (not merely the highest class conditional probability).

Using Euclidean distance and $K = 10$, the clusters converged in an average of 35.1 iterations for the 1% seed words, 44.5 iterations for the 5% seed words, and 45.4 iterations for the 10% seed words. In contrast to the purely random data, the K-means algorithm is able to find stable clusters.

As seen in Figure 4.1, the distribution of the sizes of the K clusters again follows the same pattern as the random data; that is, more than 80% of the words fall into a single cluster, leaving less than 20% in the remaining $K-1$ clusters. The words in the large cluster correspond to the great number of words that have very low frequencies in the document set.

Recall that our categories are *gas, grain, acquisitions, interest.* We know that the words with the largest differences between class conditional probability and overall probability for all *grain* articles in the Reuters collection are *tonnes, grain, agriculture, nil, department, wheat, corn, usda, crop, export*, which are the words seeded into the random seed document set at 1%, 5%, and 10% of words in each document. We want to ensure that we detect those words through our clustering process. We do obtain those words in the $K-1$ smallest clusters, but we also obtain clusters containing words with a high overall probability in the document collection.

Clusters from 1% Seeded Random

At the 1% level, for example, we get a cluster containing the grain seed words *agriculture, corn, crop, usda* with a strength of 1.0, and the additional grain seed words *department, export, grain, nil, wheat* appear in the same cluster at a strength of .4.

However, the cluster containing the grain seed words also contains 359 other words which have strength .5 or higher, most of which are not seed words in any of the four categories. We also get a very strong small cluster consisting of the words *bc, dlrs, lt, mln, pct, reute, reuter*[3]. These are words with high values of $P(w)$ (the rank order of their overall probabilities is 16, 15, 12, 11, 20, 14, 19) but which were not eliminated in the stop word list, because our stop word list is based on typical English, and not on Reuters data.

Since we know something about the domain and where these articles were obtained, and in fact have access to the entire collection, we know that "reute" and "reuter" appear in nearly all documents. Similarly, the Reuters document database deals with economic topics, and uses the special abbreviation "dlrs" for dollars. Armed with this kind of domain information, we can easily add these high frequency words to the stop word list to prevent them from being clustered **if we are restricting our analysis to Reuters articles**. It is important to note that we do *not* want to blindly eliminate the most frequently occurring words in a collection. Since we use only a small set of examples, it may be that precisely those highest frequency words in our collection are what distinguishes them from others we have not yet seen. It is only because we have access to the entire Reuters set and are restricting our analysis to that collection (for this set of experiments) that allows us to make decisions on the usefulness of the highest frequency words. Given a larger universe of discourse, the words "reute", "reuter", and "dlrs" would be those that would point us towards more Reuters economic news articles.

[3]Recall that these words are special abbreviations peculiar to the Reuters collection

In the **1%** data, using the order-dependent method of fuzzy set construction from 10 different K-means runs, if we examine the smallest $K-1$ clusters, all seed words from all categories do appear, but they are scattered throughout the $K-1$ clusters, and are not separated according to category. There are also many other non-seed words included with high strengths which reflect their high overall probability. Note that some words with very high overall probability are also seed words, such as "lt", "pct", "dlrs", "mln". We conclude that if only **1%** of a document's content contains discriminating keywords, our method will have difficulty finding them. The words are there, but so are many other words which are not discriminating. We suppose that in a solid piece of text with a reasonable length, important concept words will constitute more than 1% of the total non–stop words, so this represents a worst case scenario.

Clusters from 5% Seeded Random

In the **5%** seeded data, three clusters contained all seed words except "agriculture", which was not present in any of the $K-1$ clusters. The fuzzy clusters containing the other seed words and their strengths are given in Table 4.5 Words with strengths less than .4 are not shown.

Cluster 1 represents the words with high overall probability, including some of the seed words. Cluster 2 contains all of the seed words from the *interest* category (except "pct", which is in Cluster 1), and all but one of the seed words from the *grain* category. Cluster 3 contains all of the seed words from the *gas* category except "mln" and "pct", which are in Cluster 1. It is interesting that the seed words from

Cluster 1		Cluster 2		Cluster 3	
1.0	bc	1.0	april	1.0	bank	.9	barrels	.9	bpd
1.0	company	1.0	banks	1.0	billion	.9	crude	.9	demand
1.0	dlrs	1.0	cts	1.0	cut	.9	gasoline	.9	octane
1.0	include	1.0	fed	1.0	mar	.9	oil	.9	prices
1.0	lt	1.0	market	1.0	money	.6	shr	.6	trade
1.0	march	1.0	net	1.0	rate	.5	acquisition	.5	apr
1.0	mln	1.0	rates	1.0	stg	.5	corp	.5	loss
1.0	pct	1.0	usa	.7	corn	.5	offer	.5	rm
1.0	reute	.7	crop	.7	department	.5	share	.5	shares
1.0	reuter	.7	export	.7	grain	.5	stake	.5	stock
1.0	vs	.7	nil	.7	tonnes				
		.7	usda	.7	wheat				
		.5	apr	.5	loss				
		.5	rm	.5	share				
		.5	shares	.5	stock				
		.4	acquisition	.4	co				
		.4	offer	.4	stake				
		.4	trade						

Table 4.5: Three smallest, strongest clusters from 5% random seeded document set

the "acquisitions" category appear in both Cluster 2 and Cluster 3, with strengths of .5. This means that half the time they clustered with the *grain* and *interest* words, and half the time with *gas* words.

We do have another fairly large cluster (104 words) with strong membership (all but 9 words have strength 1.0 or .9), but which contains none of the seed words. If we restrict our focus to the smallest clusters, we find the seed words (along with some others). What we see at the 5% level is that the seed words are appearing in the smallest clusters with strong membership.

Clusters from 10% Seeded Random

In the 10% seeded data set, there are only 4 clusters that have any word at a strength greater than .6. Of these four "strong" clusters, two of them contain *all* the seed words for all four categories. These two clusters are shown in Table 4.6.

Cluster 1		Cluster 2							
gas		acquisitions		interest		grain		Non seed words	
1.0	barrels	1.0	acquisition	1.0	bank	.8	agriculture	1.0	bc
1.0	bpd	1.0	company	1.0	banks	.8	corn	1.0	reute
1.0	crude	1.0	corp	1.0	cut	.8	crop	.8	march
1.0	demand	1.0	dlrs	1.0	fed	.8	department	.8	reuter
1.0	gasoline	1.0	inc	1.0	market	.8	export	.8	vs
1.0	octane	1.0	lt	1.0	money	.8	grain	.6	mar
1.0	oil	1.0	offer	1.0	rate	.8	nil	.6	usa
1.0	prices	1.0	share	1.0	rates	.8	tonnes		
.8	mln	1.0	shares	1.0	stg	.8	usda		
.6	pct	1.0	stake	.4	pct	.8	wheat		

Table 4.6: Two clusters containing seed words from 10% random seeded document set

One cluster contained all of the *gas* seed words, and no other words. The strength of each word was 1.0 except for "mln", with a strength of .8, and "pct", with a strength of .6. This is a clear indication that we retrieved the seed words from the *gas* category. The other cluster contained all of the *acquisitions* seed words, all of the *interest* seed words, and all of the *grain* seed words. One of the other clusters containing words with strength greater than .6 contained high overall frequency words. The remaining strong cluster contained 115 words, none of which are seed words.

At a 10% level, we can detect important category words by examining the smallest clusters with strong members, and by ignoring clusters which contain words with

high overall probability. Of course, in real data situations, we would have no way of detecting which clusters are actual "seed words" (i.e., have high class conditional probabilities), and which are merely high overall probability words. But these high probability words which do not belong on the stop list are probably also important characterizing words for our example document set, even though they may not discriminate among our current categories.

4.2.3 Reuters Data Set

Next we examine the fuzzy clusters resulting from the actual Reuters data in four categories. Recall that we used the same four categories for actual data as we did for the random seeded data, and we know the class conditional probabilities for the words in each of these four categories. Therefore, we expect to obtain clusters containing those same high class probabilities words as when we "seeded" random documents with them.

The K–means algorithm converged to stable clusters quickly, in an average of 13.7 iterations. The size distributions of the clusters follow the same consistent pattern, with one large cluster containing about 80% of the words. Again, using the order dependent fuzzy cluster construction method, with $K = 10$, we obtained only 4 clusters whose strongest member was higher than .4. These clusters are shown in Table 4.7. The words of strength .6 or greater in these 4 clusters consisted almost entirely of words that are within the top 100 class conditional probabilities of one of the four target categories. For example, cluster 1 contained 35 words with strength

Cluster 1		Cluster 2		Cluster 3		Cluster 4	
.8	crude	.9	demand	.8	budget	.9	banks
.8	gasoline	.9	production	.8	co	.9	commercial
.7	companies	.9	tonnes	.8	georgia	.9	debt
.7	dunham	.7	corn	.8	power	.8	official
.7	energy	.7	crop	.8	unit	.9	world
.7	import	.7	fall	.8	vogtle	.7	brazil
.7	imports	.7	farmers	.8	board	.7	plan
.7	industry	.7	growth			.7	third
.7	unleaded	.7	soybean				
		.7	soybeans				
		.7	wheat				

Table 4.7: The four clusters from actual Reuters articles whose strongest member was greater than .4. Words with strength $< .7$ appear in the clusters, but are not shown here.

.6 or above; 19 of those 35 words appeared in the top 100 class conditional words for the *gas* category. Cluster 2 clearly indicates words from the *grain* category. The third cluster had 10 words of strength .6 or above; 4 of them were in the top 100 of the *acquisitions* category and 1 was from *interest*. The final cluster had 9 words; *banks* and *commercial* are in the top 100 class conditional words from the *interest* category, and *debt* and *official* are from the top 100 of *grain*.

These results lead us to conclude that we are successfully obtaining important keywords from the 4 categories, although the separation between categories is not always clearly delineated from the cluster information alone. We also get some extra words which do not show up in the list of the top class conditional words, especially at lower strength levels.

4.2.4 NEM–Online Data Set

The last data set is the HTML documents from NEM–Online. Here, we have no objective way to evaluate the cluster word membership, since we can not compute any class conditional probabilities. We also can not compute the overall probability for a given word; the overall probabilities for the Reuters data is not valid for this set of documents. Just as the Reuters data had words such as "reuter" and "dlrs" as high frequency words, HTML documents have words such as "homepage" and "Internet" as high frequency, non–discriminating words. We have evaluated the words in the clusters by having experts familiar with the NEM–Online domain examine them.

We ran several runs of the K–means algorithm on the NEM–Online data described above. Cluster membership stabilized in an average of 26 iterations, and again there was 1 large cluster and $K-1$ smaller ones for each run. With $K = 10$ and 10 different runs, we obtained 6 fuzzy clusters which had at least one member at strength .6 or above. These clusters are listed in Table 4.8.

The two clusters not shown here had 178 and 165 members. We note that with the NEM–Online data, as in all other data sets, we get one very large cluster (the "useless" cluster), and the remaining clusters are smaller. The smallest of these remaining clusters suggest reasonable topics, however, others are still too large for a human to easily interpret. We conjecture that the words in these larger clusters may be useful in further suggesting content if we could subdivide them into smaller pieces.

We implemented a recursive variation of the partitional K–means algorithm which accepts some user–defined size threshold, and if any cluster (other than the largest)

Cluster 1		Cluster 2		Cluster 3		Cluster 4	
.8	affirmative	1.0	amendments	1.0	cfr	.7	cad
.8	americans	1.0	bankruptcy	1.0	cosmetic	.7	cadmazing
.8	discrimination	1.0	code	1.0	cosmetics	.7	customer
.8	people	1.0	debtor	1.0	eye	.7	inc
.8	women	1.0	petition	1.0	hair	.7	management
.7	action	1.0	section	1.0	product	.7	project
.6	employee			1.0	products	.7	service
.6	fmla			.9	color	.7	services
.6	leave			.9	containing	.7	software
.4	opportunity			.9	except	.6	company
				.9	fda	.6	connx
				.9	skin	.6	consultants
						.6	consulting
						.6	customers
						.6	design
						.6	electronic
						.6	network
						.6	process
						.6	projects
						.6	solutions

Table 4.8: The four clusters from NEM–Online documents.

exceeds that threshold, the words from that cluster are reclustered according to the original documents.

Again, we have no way to objectively evaluate the worth of the words in these sub clusters. Looking at the words subjectively without the opinion of domain experts, it is not clear whether the sub categorizations are useful. Work on the hierarchical subclustering is ongoing and a subject of future study.

4.3 Cluster Usefulness

Referring to our system overview in Figure 1.1, we have described the process and results of obtaining the concept clusters. We next discuss their uses in classification accuracy and in automatic querying and document organization.

4.3.1 Classification Accuracy

Now that we have some small clusters of words, are these groups useful in document categorization? We attempted to use the resulting feature clusters to recognize the original 4 (NEM) document categories. We used a simple scheme where each of the 10 clusters was considered a feature, and an occurrence of any term from a cluster was counted as an occurrence of that feature. We used number of occurrences in each cluster as the feature values, and the 85 documents as patterns. So for example, if the phrase "affirmative action" occurred three times in a given document, and the cluster containing both "affirm" and "action" was cluster 2, the second element of the 10 dimensional feature vector for that document would contain a value of 6 (provided no other terms from that cluster also occurred in the document). If no terms from a cluster appear in a document, the value for that cluster in the feature vector is zero.

After constructing this new pattern matrix with the 10 features derived from the occurrences of terms from the 10 clusters, we performed K–nearest neighbor classification with leave one out testing, as in the feature selection cases. The classification accuracy was 42/85 errors, or 50% accuracy with $K = 1$, and didn't change significantly for other values of K.

Classification using the same scheme as described above, but with elimination of the largest cluster (therefore using only 9 features) yielded results which were only 55% accurate.

Even though the features derived from the clusters as described above did not perform very accurate classification of the original categories, we must remember that classification accuracy is not our ultimate goal. We can use the word clusters as an initial approximation of a semantic concept. We can verify the semantic relationships between the cluster words by user evaluation, or by lookup in a semantic database. We can also allow the user to refine the concept by adding or deleting words, or giving the word cluster a meaningful label (which is not necessarily the same as the user's original category labels). Our experience has shown the clusters are usually small enough that consideration of the individual words is not burdensome for a user. If a cluster is too large, the user asking the system to sub-cluster it may prove fruitful.

4.3.2 Search for Similar Documents

One of the advantages of having a semantically related group of words as a representation of a document is the ability to search for similar documents using these words. Though automatic query construction is a complicated issue, we can generate queries from word clusters using two simple strategies that show the efficacy of this approach.

A query formed by taking the *conjunction* of cluster words (for example, **crude AND gasoline AND companies AND...**, etc.) leads to a higher precision as more words are added than does a query consisting of single words alone. However,

recall will decrease as words are added to the conjunction. A query formed by the *disjunction* of cluster words (for example, `crude OR gasoline OR companies OR...`, etc.) results in a higher recall, but lower precision as more words are added to the query.

Remember from Table 4.7 that cluster 1 words suggest the category "gas" and cluster 2 words suggest the category "grain". We used these as target categories for purposes of determining document relevance.

Using the 9 words of cluster 1 with strength .7 or greater, and the 11 words of cluster 2 with strength .7 or greater, we generated all possible conjunctive queries of all sizes for the target categories "gas" and "grain". We computed precision and recall for each query as defined earlier. For purposes of this experiment, "Retrieved" documents were all those that satisfied the conjunction and "Relevant" were those that carried the same label as the target category. Note that this definition of "relevant" may not be correct, due to the multiple, inconsistent, and erroneous labelings in the Reuters data. However, we used that definition of relevance to provide as objective a measure as possible.

The maximum precision and recall values for a given size query for the two concept clusters are shown in Figure 4.2 and Figure 4.3. We see that, as expected, for conjunctive queries the precision increases with additional words.

We also simulated disjunctive queries, where we are interested in improving recall. We know that many disjunctive terms will decrease precision since a single word from the query occurring in a document is enough to retrieve that document. We generated queries consisting of the disjunction of *all* words in Reuters cluster 1 and cluster 2

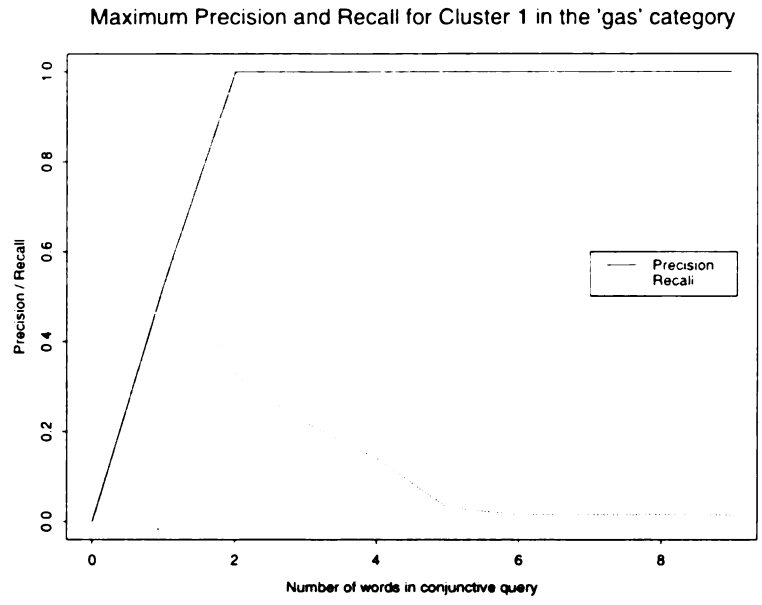

Figure 4.2: Maximum Precision and Recall for Reuters cluster 1 in the gas category

(Table 4.7), and measured the percentage of documents retrieved in each of our 4 target categories. Results are shown in Table 4.9.

Category	Cluster 1	Cluster 2
	Percentage retrieved	Percentage retrieved
acquisitions	.16	.07
gas	.75	.39
interest	.12	.29
grain	.25	.86

Table 4.9: Occurrences of cluster words in all Reuters documents of the four categories

We see that the disjunction of cluster 1 words retrieved the highest percentage of "gas" documents, and the disjunction of cluster 2 words retrieved the highest percentage of "grain" documents from among the four targeted categories.

Precision and recall of single word queries of the target categories, where the query consisted of the obvious choice of category label are given in Table 4.10.

For example, the single word query "gas" for the target category "gas" resulted in

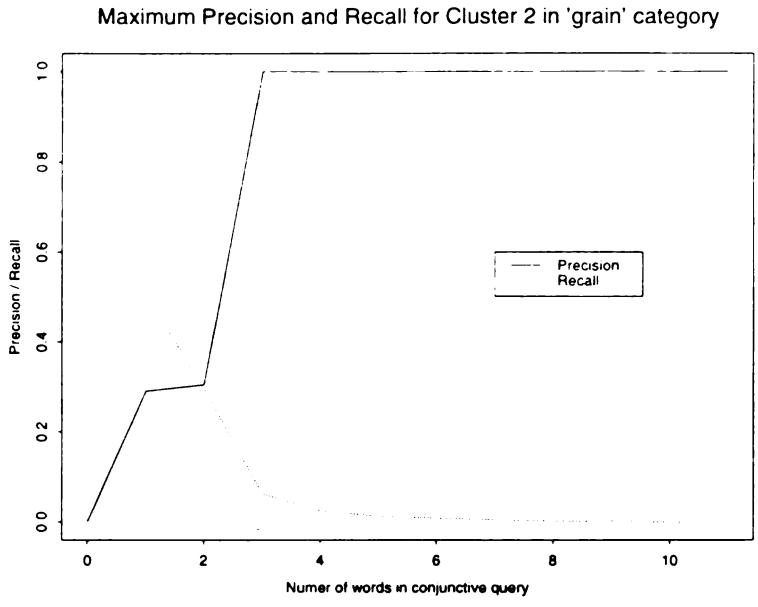

Figure 4.3: Maximum Precision and Recall for Reuters cluster 2 in the grain category

Category	Query Term	Precision	Recall
acq	acquisition	.80	.22
gas	gas	.03	.21
interest	interest	.15	.37
grain	grain	.31	.39

Table 4.10: Precision and Recall of single word queries

precision of .025 and recall of .21. The best precision single word query formed from the concept clusters in the category "gas" was the word "unleaded", with a precision of .52. For the query "grain" in the "grain" category, precision was .31 and recall was .39. A single word query from cluster 2 consisting of "corn" and another query of "wheat" both exceeded the recall of the "grain" query. So even without constructing multiple word queries, word clustering can obtain single word queries which perform better than the category labels themselves.

If we then examine queries consisting of the conjunction of 2 words, we see that the query "tonnes AND wheat" achieved precision of .30, and recall of 30. The three

word queries "tonnes AND farmers AND growth", and "corn AND farmers AND growth" achieved 1.0 precision.

We know that most web queries consist of one or two words [24]; we see that by using more words per query we can improve precision with conjunction and recall with disjunction. Furthermore, by providing a set of words, we can achieve greater performance in both precision and recall by suggesting words a user may not have thought of (e.g., "unleaded"). More complicated combinations of operators for query generation may be able to improve both precision and recall simultaneously by using the words from the feature clusters and their strengths. This is a subject for further research.

4.3.3 Personalized Document Organization

Another application for our concept clusters of words is for organizing a collection of documents (listed as "organization with topical proximity" in Figure 1.1). We have already seen that classification of documents into the original training set categories does not work well. Instead, we used an unsupervised clustering method to group the original documents using our derived concept clusters as features. Recall that in our original pattern matrix M, entry $M_{i,j}$ represented the number of times word j occurred in document i. We now form a new $i \times c$ matrix P, where rows represent documents, and columns represent fuzzy clusters. We define an entry

$$P_{i,j} = \sum_{k=1}^{n} w_k M_{i,k}$$

where n is the number of words in fuzzy cluster j, and w_k is the fuzzy membership value (i.e., the strength) of word k in cluster j.

We then cluster the *rows* of P using K-means partitional clustering and Euclidean distance. We now have many documents but only a few features.

If we examine only the original *labels* of the documents to evaluate our clustering, we see that small, tight clusters have the same labeling, but there are also many documents that do not go into a small cluster. These are then combined into one cluster whose members have several category labels. However, if we look closely at the *content* of the clustered documents rather than just their labels, we see some close similarities between the documents. For example, for the NEM data, we see that a group of 3 hardware/software/electronics consulting companies always clusters together. The text of the Family Medical Leave Act (FMLA) occurs along with a document entitled "11 easy steps to avoid FMLA problems", "Employee relations: Resources on Affirmative Action", "Judy's affirmative action page", and "President's remarks on affirmative action". We also get a cluster containing the 3 documents "Poisonous substances in human and animal feed", "FDA import procedures", and "FDA recall policies". Another contains "Trusts and Estates bulletin", "GATT related changes in the U.S. Patent System", "Overview of the U.S. Patent", "Preparing the defective product case", and "Defective product litigation". While we did not get neat clusters of our original categories back, we did get clusters of documents that are more closely related to each other than the broad topics defined by the original labels.

The Reuters articles were less clear, although a similar argument can be made for

closely related articles getting grouped together. For example, one cluster contained 4 documents, 2 each from the categories "acquisitions" and "interest". However, if we examine those 4 documents, one of the "acq" articles is about Brazil's bank debt. In another cluster, there were 7 "interest" documents and one "acquisitions". Here again, the "acquisitions" article was about Bank of America's German branch selling Bankhaus Centrale Credit, with words such as "bank" and "credit" occurring several times, lending credence to the idea that it should be together with articles relating to banking which carry the label "interest". That same cluster also revealed two identical articles under different identifiers, which clustered together.

There were no small clusters among the Reuters set that contained the "gas" label. Upon inspection of our 20 randomly selected "gas" articles, it was discovered that some of them were mis-labeled (articles about "the decline in gold production", "Vermont Financial Services approved cash dividend to shareholders", and "debt securities offered for sale"), and the remainder varied widely in their subject matter, though they could be reasonably considered to be in the category "gas". For example, there were articles about ethanol tax exemption, energy costs rising, the Ford Escort passing European emissions tests, the EPA proposing rules to shield water supplies from leaks of underground tanks, and a company proposing a plant to build an octane enhancer to replace lead. Though all can be considered to fall under the category "gas", they are not very closely related to each other, rather they represent many subtopics under "gas". The 20 "gas" articles chosen (at random) were not closely related to each other, however, in spite of that we did get a word cluster with words like *crude, gasoline, energy, unleaded, diesel, octane, petroleum, refining* which suggest the

concept of gas. This brings up an important point about document categorization. The process of assigning documents to categories is difficult even for humans, except in the most clear cut and obvious cases. We should not expect that a computer will be able to categorize documents according to meaning which is not explicitly represented, when humans often have difficulty with the same task.

4.3.4 Conclusions

In this chapter we have described our experimental results in deriving concept clusters. We demonstrated that random documents (which nevertheless had the same distribution of words as the entire document collection) did not converge into stable clusters of semantically related words. We showed that the random documents which were seeded with category words obtained small, strong clusters containing those seed words. We also showed that for actual documents, the small strong clusters contained words which had a large difference between their class conditional probabilities and their overall probabilities. The categories were not well separated by the concept clusters; words from different clusters often appeared together, but were still present in the small clusters. We conclude that when we use our method on real documents, we obtain words which have a high difference between their class conditional probabilities and their overall probabilities, or, words which have high overall probability within that set but were not removed by a stop list.

We can use the concept clusters formed from our method as a basis for solving other problems such as search for documents on similar topics, and organization of

document collections according to personal categorization schemes.

Chapter 5

Summary and Future Work

In this chapter we summarize the results of the research reported in this dissertation, and discuss a number of directions for future research.

5.1 Summary

Using only statistical techniques, we have developed a method of finding conceptually related groups of words which represent the underlying content of user–designated text examples. We use these groups of words as a representation of documents that is at an intermediate conceptual level between single word representations and global category representations.

We assume that a user has collected several documents from each of several categories of interest so that the document collection contains both commonality between documents in the same category, and diversity among the categories. This is a reasonable expectation of the user. We also assume that the number of example documents

is smaller than the number of distinct words contained in all documents. If there are more documents than words, and if we attempt to use the entire document set to obtain word clusters, we are likely to suffer the problem of of a sparse high dimensional space that was discussed in Chapter 3. In that case, we would need to break the document collection into subsets (subspaces of lower dimension) which meet the diversity and commonality requirements. The method described here was intended for personal use on few example documents, therefore all processing should take place in a reasonably short time.

Another assumption is that the documents in the same category share a common vocabulary which denotes the category. Users often categorize documents in this way (e.g., documents about vacations vs. documents about job openings), but also sometimes categorize documents according to meta concepts like humor or color schemes. When these documents in the same category do not share common vocabulary which distinguishes them from other categories, our method will fail. The documents need not be labeled by category, it is enough that the categories are present implicitly.

We described the limitations of word-level and global representations of documents, and proposed an intermediate conceptual level of representation of text using our groups of semantically related words. This is the major contribution of this thesis.

We described our initial attempts at finding good features from among all the words in documents, using classification accuracy as a performance measure. We concluded that when the number of examples is as small as those we are working with, classification accuracy is not a good measure of the worth of features. Traditional feature selection and extraction techniques obtained highly accurate classification,

but features which do not convey semantic content of the documents. Feature selection and extraction techniques use a word-level representation of a document, but classification accuracy is a global representation. Using one extreme of representation as a measure of the worth of the other extreme is not a good idea, especially when the goal is content representation, and not classification. This realization is a second contribution of this thesis.

We used word clustering to obtain groups of semantically related words which *do* convey content information. We reported on using Euclidean distance, cosine distance, and Jaccard's coefficient. Cosine distance is the one most favored by researchers in the information retrieval community. We concluded that while cosine distance makes sense as a measure of distance between *documents*, it does not make sense as a measure of distance between *words*.

We discussed normalization for document length and concluded that it is not necessary, and may in fact be harmful, although this is an area for future study.

Next we described a method for combining the clusters resulting from different runs of the clustering algorithm into fuzzy sets whose members have a strength function according to their proclivity to co-occur with other members of the group. We described two different algorithms for determining the fuzzy set membership function; one which depends on the order in which the results are presented but is fast, and one which is not order dependent but which is computationally resource intensive.

In chapter 4 we presented results of experiments on 4 different data sets, and demonstrated that we can find words with high class conditional probabilities without many training samples, and without needing to explicitly compute those probabilities,

or the overall word probabilities. This is an important result. We also described the uses of the word concept clusters in classification, search, and document organization. We concluded that although using the concept clusters for classification into the original categories was not very accurate, the concept clusters are useful in searches and document organization.

5.2 Future Work

There are a number of different directions for future research, which we describe below.

5.2.1 Characterization of Training Set

By using the "seed" words from the Reuters document collection, we showed that we can find the words with high class conditional probabilities, as long as the number of concept words is somewhere between 5 and 10% of the documents. We used a fixed document size, fixed seed word set size, and a fixed number of categories. There are many more studies that can be done to evaluate the effect of these parameters on the success in finding the appropriate words. Then we can determine necessary characteristics of the example document set.

One possibility is to have a larger seed word set size, but seed each document with only a fraction of that seed word set. We could see whether we can still detect the entire set, and if so what portion of the seed word set needs to be included in each document in order to successfully find the seed words.

Varying the number of classes would allow us to analyze performance with respect to the commonality / diversity of the training set. The experiments reported here used 4 classes which were for the most part disjoint. It would be interesting to see if smaller numbers of categories or larger number of categories affect the ability to detect the key words. It would also be interesting to see whether including a reject class (that is, a set of random documents with no seed words) would improve, hinder, or not affect the formation of concept clusters.

We could also study the effect document length has on the resulting clusters. We have already hypothesized, based on empirical evidence, that since we do not normalize for document length, we need documents of a certain size in order to extract the correct context and thus the correct concept clusters. We would like to see how short documents can be and still yield concept clusters.

5.2.2 Hierarchical Subclustering

We have implemented a hierarchical version of the K–means clustering algorithm which subclusters any cluster which exceeds a user–defined threshold. We have not fully studied the effects of the sub–clustering, and a next step would be to determine whether we can derive sub–concepts by hierarchically clustering, and if so, what characteristics of the training examples are necessary to achieve meaningful subclusters.

5.2.3 Other methods for determining fuzzy set membership

As we mentioned, creating the fuzzy clusters by building the full co-occurrence matrix and traversing it several times according to different strength levels is computationally complex. It would be interesting to try different methods of determining the "strength" of a word in the cluster. Some possibilities for the strength function $strength(w)$ in cluster C were mentioned in section 3.4.4. The challenge is to develop an algorithm that is efficient enough for use by average users at any time.

5.2.4 Applications for conceptual representation

There are many directions to investigate regarding automatic search for new documents using a word cluster representation. We could run our own indexing spider which indexes documents according to the concept clusters, and then use it for retrieval with the automatic queries generated from the concept clusters.

There is also great research potential in studying automatic query generation. One interesting idea is to use genetic programming to construct Boolean queries using cluster words. Not only could these queries lead to improved retrieval of documents, but a further representation of the information contained therein.

5.2.5 Incorporate knowledge in representation

Perhaps the most interesting direction for future research involves incorporating *knowledge* into creation of the concept features rather than just statistics of single words. We can use a semantic database such as WordNet [71] to determine relation-

ships between the concept cluster words. We could also use word pairs for phrase detection, or natural language processing techniques to further determine word relationships. Techniques which were computationally prohibitive when considering *all* words in a document are no longer intractable when working with the concept clusters. Incorporating knowledge will allow further abstraction of information beyond simple *sets* of individual words, which we have dealt with in the research reported in this dissertation. Sets of words with relationships such as "follows" (phrase formation), "islike" (synonyms), "isakindof" (subclass), etc. offers a potentially more powerful representation of information.

Bibliography

[1] G. Salton, *Automatic Information Organization and Retrieval.* McGraw-Hill, 1968.

[2] M. Dewey, *Abridged Dewey decimal classification and relative index.* Lake Placid Club, NY: Forest Press, 10th ed., 1971.

[3] L. M. Chan, *Library of Congress Subject Headings: Principles and Application.* Englewood, CO: Libraries Unlimited, 1995.

[4] Yahoo. [Online] Available http://www.yahoo.com, July 31, 1998.

[5] Infoseek. [Online] Available http://www.infoseek.com, July 31, 1998.

[6] Lycos. [Online] Available http://www.lycos.com, July 31, 1998.

[7] AltaVista. [Online] Available http://altavista.digital.com, July 31, 1998.

[8] WebCrawler. [Online] Available http://www.webcrawler.com, July 31, 1998.

[9] Opentext (Livelink Pinstripe). [Online] Available http://pinstripe.opentext.com, July 31, 1998.

[10] M. Koster, "The web robots pages." [Online] Available http://info.webcrawler.com/mak/projects/robots/robots.html, July 31, 1998.

[11] W. B. Frakes and R. Baeza-Yates, eds., *Information Retrieval Data Structures and Algorithms.* Englewood Cliffs, New Jersey: Prentice Hall, 1992.

[12] J. D. Martin, "Clustering full text documents," in *Proccedings of IJCAI-95 workshop on Data Engineering for Inductive Learning*, 1995.

[13] M. Pazzani, L. Nguyen, and S. Mantik, "Learning from hotlists and coldlists: Towards a www information filtering and seeking agent," in *Proc. AI Tools Conf*, (Washington, DC), 1995.

[14] G. Salton and M. McGill, *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[15] C. J. van Rijsbergen, *Information Retrieval.* London: Butterworth & Co, 2nd ed., 1979.

[16] A. Singhal, C. Buckley, and M. Mitra, "Pivoted document length normalization," in *Proceedings of the Nineteenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, (Zurich, Switzerland), pp. 21–29, August 1996.

[17] H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM Journal of Research and Development*, vol. 1, no. 4, 1957.

[18] C. Fox, "Lexical analysis and stoplists," in *Information Retrieval Data Structures and Algorithms* (W. B. Frakes and R. Baeza-Yates, eds.), pp. 102–130, Englewood Cliffs, New Jersey: Prentice Hall, 1992.

[19] W. B. Frakes, "Stemming algorithms," in *Information Retrieval Data Structures and Algorithms* (W. B. Frakes and R. Baeza-Yates, eds.), pp. 131–160, Englewood Cliffs, New Jersey: Prentice Hall, 1992.

[20] A. Bookstein, S. T. Klein, and T. Raita, "Clumping properties of content-bearing words," *Journal of the American Society for Information Science*, vol. 49, pp. 102–114, February 1998.

[21] S. Deerwester, S. T. Dumais, G. W. Fumas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[22] S. K. M. Wong, W. Ziarko, V. V. Raghavan, and P. C. N. Wong, "On modeling of information retrieval concepts in vector spaces," *ACM Transactions on Database Systems*, vol. 12, pp. 299–321, June 1987.

[23] G. Z. Liu, "Semantic vector space model: Implementation and evaluation," *Journal of the American Society for Information Science*, vol. 48, pp. 395–417, May 1997.

[24] W. B. Croft, R. Cook, and D. Wilder, "Providing government information on the internet: Experiences with thomas," in *Digital Libraries Conference DL'95*, (Austin, TX), pp. 19–24, 1995.

[25] D. Harman, "Relevance feedback and other query modification techniques," in *Information Retrieval Data Structures and Algorithms* (W. B. Frakes and R. Baeza-Yates, eds.), pp. 241–263, Englewood Cliffs, New Jersey: Prentice Hall, 1992.

[26] K. S. Jones, *Automatic Keyword Classification for Information Retrieval.* London: Butterworth, 1971.

[27] K. S. Jones and E. O. Barber, "What makes an automatic keyword classification effective," *Journal of the American Society for Information Science*, vol. 22, no. 3, pp. 166–175, 1971.

[28] J. Minker, G. A. Wilson, and B. H. Zimmerman, "An evaluation of query expansion by the addition of clustered terms for a document retrieval system," *Information Storage and Retrieval*, vol. 8, no. 6, pp. 329–348, 1972.

[29] D. Harman, "Towards interactive query expansion," in *Proceedings of the Eleventh Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, (Grenoble, France), 1988.

[30] R. Attar and A. S. Fraenkel, "Local feedback in full–text retrieval systems," *Journal of the Association for Computing Machinery*, vol. 24, no. 3, pp. 397–417, 1977.

[31] W. B. Croft and D. J. Harper, "Using probabilistic models of document retrieval without relevance information," *Journal of Documentation*, vol. 35, pp. 285–295, 1979.

[32] J. Xu and W. B. Croft, "Query expansion using local and global document analysis," in *Proceedings of the Nineteenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.* (Zurich, Switzerland), pp. 4–11, August 1996.

[33] C. J. Crouch and B. Yang, "Experiments in automatic statistical thesaurus construction," in *Proceedings of the Fifteenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, (Denmark), pp. 77–87, 1992.

[34] Live Topics help page, http://altavista.digital.com/av/lt/help.html.

[35] C. Faloutsos and D. Oard, "A survey of information retrieval and filtering methods," Tech. Rep. TR-CS-3514, University of Maryland, oard@glue.umd.edu, August 1995.

[36] Y. Y. Yao, "Measuring retrieval effectiveness based on user preference of documents," *Journal of the American Society for Information Science*, vol. 46, pp. 133–145, March 1995.

[37] M. Marchiori, "The quest for correct information on the web: Hyper search engines," in *Sixth International World Wide Web Conference*, (Santa Clara, CA), April 1997.

[38] M. Slot, "The matrix of internet catalogs and search engines," 1996. http://www.ambrosiasw.com/ fprefect/matrix/.

[39] D. Sullivan, Mar 1998. http:searchenginewatch.internet.com.

[40] H. Lieberman, "Letizia: An agent that assists web browsing," in *Proceedings of the International Joint Conference on Artificial Intelligence*, (Montreal), 1995.

[41] M. Balabanović, "An adaptive web page recommendation service," in *First International Conference on Autonomous Agents*, (Marina del Rey, CA), February 1997.

[42] M. Balabanović and Y. Shoham, "Learning information retrieval agents: Experiments with automated web browsing," in *Proc. 1995 AAAI Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments*, (Stanford), March 1995.

[43] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell, "Webwatcher: A learning apprentice for the world wide web," *Proceedings of the 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, March 1995.

[44] S. Cost and S. Salzberg, "A weighted nearest neighbor algorithm for learning with symbolic features," *Machine Learning*, vol. 10, pp. 57–78, 1993.

[45] T. Honkela, V. Pulkki, and T. Kohonen, "Contextual relations of words in grimm tales, analyzed by self–organizing map," in *Proceedings of International Conference on Artificial Neural Networks, ICANN–95* (F. Fogelman-Soulie and P. Gallinari, eds.), (Paris), pp. 3–7, 1995.

[46] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen, "Newsgroup exploration with websom method and browsing interface," Tech. Rep. A32, Helsinki University of Technology, Laboratory of Computer and Information Science, January 1996.

[47] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen, "Exploration of full-text databases with self-organizing maps," in *Proceedings of International Conference on Artificial Neural Networks, ICANN-96*, (Washington D.C.), 1996.

[48] S. Kaski, T. Honkela, K. Lagus, and T. Kohonen, "Creating an order in digital libraries with self-organizing maps," in *Proceeings WCNN'96, World Congress on Neural Networks*, (Mahwah, NJ), 1996.

[49] R. Weiss, B. Vélez, M. A. Sheldon, C. Namprempre, P. Szilagyi, A. Duda, and D. K. Gifford, "Hypursuit: A hierarchical network search engine that exploits content-link hypertext clustering," in *Hypertext'96: The Seventh ACM Conference on Hypertext*, (Washington D.C.), March 1996.

[50] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic clustering of the web," in *Sixth International World Wide Web Conference*, (Santa Clara, CA), April 1997.

[51] G. A. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, pp. 39-41, November 1995.

[52] M. Sahami, S. Yusufali, and M. Q. W. Baldonado, "Sonia: A service for organizing networked information autonomously," in *Digital Libraries 98: Proceedings of*

the Third ACM Conference on Digital Libraries, (New York, NY), pp. 200–209, 1998.

[53] P. Pirolli, P. Schank, M. Hearst, and C. Diehl, "Scatter/gather browsing communicates the topic structure of a very large text collection," in *Human Factors in Computing Systems CHI '96*, pp. 213–220, Association for Computing Machinery, 1996.

[54] H. Schutze and C. Silverstein, "Projections for efficient document clustering," in *Proceedings of the Twentieth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, (Philadelphia, PA), pp. 74–81, 1997.

[55] C. Silverstein and J. O. Pederson, "Almost-constant-time clustering of arbitrary corpus subsets," in *Proceedings of the Twentieth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, (Philadelphia, PA), pp. 60–66, 1997.

[56] M. A. Hearst and J. O. Pedersen, "Reexamining the cluster hypothesis: Scatter/gather on retrieval results," in *Proceedings of the Nineteenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, (Zurich, Switzerland), pp. 76–84, August 1996.

[57] Y. S. Maarek and I. Z. B. Shaul, "Automatically organizing bookmarks per contents," in *Fifth International World Wide Web Conference*, (Paris, France), May 1996.

[58] A. K. Jain and B. Chandrasekaran, "Dimensionality and sample size considerations in pattern recognition practice," in *Handbook of Statistics, Vol. 2* (P. R. Krishnaiah and L. N. Kanal, eds.), pp. 835–855, Amsterdam: North–Holland Publishing Company, 1982.

[59] A. Whitney, "A direct method of nonparametric measurement selection," *IEEE Transactions on Computers*, vol. 20, pp. 1100–1103, 1971.

[60] F. J. Ferri, P. Pudil, M. Hatef, and J. Kittler, "Comparative study of techniques for large–scale feature selection," in *Pattern Recognition in Practice IV, Mutliple Paradigms, Comparative Studies and Hybrid Systems* (E. S. Gelsema and L. S. Kanal, eds.), pp. 403–413, Amsterdam: Elsevier, 1994.

[61] W. F. Punch, E. D. Goodman, M. Pei, L. Chia-Shun, P. Hovland, and R. Enbody, "Further research on feature selection and classification using genetic algorithms," in *International Conference on Genetic Algorithms 93*, (Champaign, IL), 1993.

[62] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large scale feature selection," *Pattern Recognition Letters*, vol. 10, pp. 335–347, November 1989.

[63] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, Michigan: University of Michigan Press, 1975.

[64] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, Massachusetts: Addison–Wesley, 1989.

[65] D. L. Swets and J. J. Weng, "Using discriminant eigenfeatures for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, October 1996.

[66] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, New Jersey: Prentice Hall, 1988.

[67] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[68] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[69] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press, 1981.

[70] H. P. Zipf, *Human Behavior and the Principle of Least Effort*. Cambridge, Massachusetts: Addison–Wesley, 1949.

[71] G. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, pp. 39–41, Nov 1995.