SIFTING AND WINNOWING: APPROACHES TO FINDING USEFUL INFORMATION ON THE WEB

By

Matthew Zeidenberg

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN – MADISON

2003

Contents

List of Tables

List of Figures

Abstract

The Web has started a social transformation in which information flows more widely and could be made more reliable. But we need tools to make this flood of information more useful.

I argue that web search engines, directories, and collaborative filtering systems should be combined into unified systems based on the general concept of *information filtering*. Such filtering can be used with Web pages, user reputations, and anything subject to a price or a poll. An information filtering mechanism can supplement or perhaps supplant markets and other institutions.

Computers are best at processing large amounts of textual information quickly and making a good first guess on such attributes of a page as what category it belongs in, what other pages it is close to, and its rank relative to the other pages. Communities supporting user reputations are best at final judgments.

A system that combines a Web directory and search engine can be built by spidering off the initial directory pages and using a multi-resolution version of the Naive Bayes algorithm that classifies the new pages in a scalable manner. I also show that links can add valuable information about what category in which a page belongs.

By collecting human judgments about a set of pages, I find only a weak relationship between these judgments and a count of in-links to these pages. If human judgments are what one is looking for, there is no way to get such judgments that is better than gathering them directly, through collaborative filtering.

I experiment with various ways of clustering web pages, first by using links and pruning out highly-referenced pages, second by using the WordNet electronic lexicon, and third by using semantic networks constructed from the document text. All of these methods are successful, to varying degrees.

I build a linear system for collaborative filtering with an explicit relation between document and author. Ratings of documents reflect on their authors, and these influence the weight given to their rating of other documents. Such a system is shown to be stable after relaxation.

Acknowledgements

I would like to thank Jude Shavlik for taking over the advising of this thesis after the untimely death of Len Uhr. Jude put in considerable time in helping me improve the quality of this thesis. I would also like to thank the other members of my thesis committee, Chuck Dyer, Vasant Honavar, Mark Craven, and David Page. I would also like to thank the other faculty in the Computer Sciences department who taught me, especially Gregg Oden (now of the University of Iowa), who served as my first advisor. I would also like to thank my fellow students and friends in the Computer Sciences Department from whom I learned a great deal, including Randy Whitaker, John Janetos, and Ganesh Mani.

I would like to thank my family for giving me a lot of support, including my parents Rayna and Phillip, my sisters Lisa and Debbie, my grandmother Toby (who passed away about six months before I completed my thesis), my brother-in-law Evan, and my niece Sasha. I would also like to thank the many friends who supported me, especially Karen Helfand-Bix, George Tsibouris, Kenn Kamau, Duncan Chaplin, Bonnie Farber, Kate Wheeler, Lawrence Cohen, Igor Steinberg, Kassie Remo, Sengbeng Ho, Jessica Ward Lynch, and Joslyn Cassady. Although she passed away almost nine years ago, my friendship with Carolyn Rosner continues to have a strong positive effect on my life.

I would like to thank Joel Rogers and Laura Dresser and the rest of my friends among the staff at the Center on Wisconsin Strategy for providing longtime employment and for their friendship and fellowship.

I would like to thank Bob Factor, Chuck Heikkinen, Jeff Hird, and the members of my dissertation support group for their support. I especially want to thank Tamar Zick–I'm not sure I would have ever finished this thesis without her help.

I would like to thank Jo Ann Oravec for providing the student-subjects used in Chapter 5, and for supervising the running of the experiment involving those subjects. Thanks also to Mina C. Johnson-Glenberg for introducing me to the HAL model used in Chapter 6.

I would like to thank the members of the Open Source community for developing many of the tools that I made heavy use of in writing this thesis, including Linux, Perl, TEX, LATEX, LYX, wget, lynx, and the Open Directory.

I would like to dedicate this thesis to the memory of two people who had a major impact on my life. My beloved grandmother, Mrs. Toby Schwartz, passed away last fall at the age of 100. Many Scrabble games with her over the years honed both the verbal and spatial sides of my mind, and she always implicitly conveyed the love of learning that is fortunately fairly common within the Jewish community. She supported me for years by being continuously proud of me as her grandson.

Professor Leonard M. Uhr, my longtime advisor, was taken from us unexpectedly in late 2000. Len was a dedicated scholar and someone who was always trying to see the "big picture," and always pressed me to go beyond technical details of algorithms to keep in mind the larger purposes of what I was trying to do, and to grasp things at a more general level. I think that this perspective comes through strongly in this thesis. Len is sorely missed.

Chapter 1

Introduction

Whatever may be the limitations which trammel inquiry elsewhere, we believe that the great state University of Wisconsin should ever encourage that continual and fearless sifting and winnowing by which alone the truth can be found. (Statement of the Board of Regents, University of Wisconsin, 1894)[1]

1.1 Information Filtering

As anyone who has not spent the last several years with their head, ostrich-like, buried in the sand knows, the World Wide Web and its substrate or super-set, the Internet, have been growing by leaps and bounds over the last several years. Since the Web is democratic (in the sense that almost anyone, at least in countries in the developed

[1]This statement, which is found on a plaque outside Bascom Hall, the main administrative building of University of Wisconsin-Madison, was the result of an interesting battle over academic freedom. For more details, see the Wisconsin Electronic Reader at http://www.library.wisc.edu/etext/WIReader/Contents/Sifting.html.

world, can create a Web page or site[2]) and decentralized, it tends to be rather chaotic and disorganized. People searching the Web have difficulty finding useful information, and often have to sift through large numbers of relatively useless pages in order to find the information for which they are looking.

The purpose of this thesis is to explore how the Web might be better organized through the use of various manual, collaborative, and automatic techniques. In addition, I discuss how such improved organization could be used in service of improving the quality and productivity of people acting together in communities. Organizational techniques that have been used by researchers and practitioners include: unsupervised page clustering based on a page similarity metric (e.g. [118][126][130]), manual page classification (as practiced by Yahoo! or About.com), automatic page classification based on learning from manually classified pages (e.g. [60][80]), accessing page content and significance via an examination of the hyper-link structure of the Web (e.g. [14][21]), and accessing page relevance and page quality via user feedback and user modeling (e.g. [8][112]).

The argument of this thesis is that the concept of *information filtering* [81] can generalize the techniques commonly used to organize information on the Web. I define information filtering as a general process whereby information on the Web is selected, ranked, and organized for presentation to a user. The information is filtered on the basis of a user query or other user interaction (such as selecting from a menu). In this thesis, I will attempt to show that techniques such as search engines, semantic networks, directories, and collaboration systems can be combined so as to

[2]Although one can argue that it is not any more democratic than any other communications medium, because those with more financial or other social resources can gain much more attention on the web than those who do not, in many cases, even within the relatively wealthy countries.

be likely to produce a better organization of Web pages than any of these techniques on their own. All of these systems act as filters of one kind or another, usually using different information to do their filtering. Link information alone is unlikely to disentangle pages that are thematically different but linked together; semantic content information is likely to be needed to separate pages into their categories. Web sites and Web pages can be usefully said to have the following (non-exhaustive) list of properties:

- How popular they are
- How useful or informative they are (this is relative to the individual that is using them and the particular task that that individual has in mind, although one could also devise a concept of general usefulness or informativeness). This may be measured by user ratings or by document analysis techniques designed to measure the specific content of a document.
- How many other sites and pages link to them (which is related to the first item above, their popularity; if this is high the Clever project [58] calls the page an "authority;" links may also relate to usefulness)
- How many other sites and pages they link to (if this is high the Clever project calls the page a "hub").
- What they are about (this is similar to their classification in a library card catalog); note that any particular page or site, like any particular book or article, may have several classifications

- Where they are in the hyperspace of the Web; that is, what neighborhood they are in relative to other sites. So, for instance, sites about art museums tend to be closer to sites about Picasso than sites about mountain biking.
- What institution they are associated with (if any), and where this institution is physically located
- What people are associated with the page, as authors of the page or as individuals described on the page

Search engines and agents that hope to provide useful information to people who seek it need to compile much of the above information about Web sites and pages in order to help filter the sites and pages. One heuristic might be the following: if a user makes a low rating of a particular page on a site, the engine or agent should adapt its behavior, giving a low ranking to other pages on that site, on the assumption that if the site has one lousy page, it is likely to have others, if they are all authored by the same person or institution. Particular institutions or individuals may become, in the view of users, high or low quality providers of information. However, if a page from a previously low rated site appears to have a large amount of content on it, perhaps measured by the presence of a lot of specific words on the page, than that page may be promoted so that the user may make a second evaluation. More recent evaluations should be weighted higher than older ones.

A particular query to a search engine or directory constitutes a partial description of a particular search task; that is, the person composing the query usually has a better idea of what he or she is looking for than is embodied in the query. That is, the searcher is typically not good at expressing her needs in terms of a query. For instance, I was interested in whether or not HDTV (high-definition TV) programming would be available on DVD media, so I typed the query "+HDTV +DVD" into Yahoo! to try to find pages that contained both concepts. If the search engine were smart, I could have told it what I wanted to know in natural language, and it would be able to find a page that answered the specific question I had, if such a page existed. Much research has gone into trying to understand natural language queries, and little progress, in terms of the development of practical, deployed systems, has been made outside of tightly-defined topic areas.[3] Currently, approaches to organizing information on the Web can mainly be divided into the following categories:

1. Classification into categories (such as done by systems like Yahoo!—at www.yahoo.com—or the Open Directory Project—at www.dmoz.org) [83]. Typically, such systems organize Web pages into hierarchies, in a manner similar to a card catalog in a library. These projects are very intensive in terms of human labor. Ironically, very little automation has been used in the construction of such hierarchies, beyond simple utilities for the insertion of links and maintenance of hierarchies. I will suggest ways in which additional automation can vastly improve the construction of such hierarchies, while at the same time making them more useful.

2. Organization created implicitly by search engines such as Google—at www.google.com—or AltaVista—at www.altavista.com—which build inverse keyword indexes of pages that they gather by spidering the Web. Each keyword or

[3]I tried the "hdtv dvd" query on Ask Jeeves (at www.ask.com), the best known of the natural-language based search engines, and it came up with five questions and answers, none of which were what I was looking for. The question "When will HDTV movies be available on DVD?" fared no better. It appears that Ask Jeeves only works well when the question entered matches one of the case frames that the system already knows about.

logical expression constructed out of keywords implies a set of pages that are implicitly organized into a class. The system then has the problem of ranking the documents in the query result; often some combination of the in-links to each page and the frequency of occurrence of the search keywords on each page in the list. Often, searchers—especially those with little searching experience—find it difficult to use search engines to locate the information for which they are looking.

3. Manual "hub" pages built by individuals or corporate entities gathering together a number of links on a specific topic or a set of closely-related topics.

4. Pages that contain collectively-maintained definitive information about a particular topic, such as frequently-asked-question (FAQ) files. Such pages often have many other pages pointing to them, and have been referred to in the literature as "authorities."

5. Organization created by collaborative filtering of information items [44, 108]. In collaborative filtering, user preferences are collected and aggregated, and users are matched with users with similar interests using matching algorithms, such as simple correlations of their preference vectors.

   Some of the best known examples of these are the GroupLens system [108], which collaboratively filters Usenet news articles, and the MovieLens system [28], which collaboratively filters movies. There are emerging systems for collaboratively filtering postings on Web sites, such as the system used by the popular Slashdot Web site (at www.slashdot.org), and the system used by Advogato (at www.advogato.org), which uses rings of trust modeled on those used in cryptography, to build up hierarchies among open source software developers. However, collaborative filtering systems suffer from difficulties in collecting sufficient ratings to make them work well. I believe that there are ways that existing social networks may be exploited in order to overcome this problem.

Note that it is important to recognize that the rankings that Web search engines put on pages are really just a poor substitute for the rankings that could be gathered using collaborative filtering systems. Web pages become just of the one of the many entities in the world, physical, institutional, and informational, that could be filtered by communities acting collectively.

6. Organization derived from knowledge of the semantic structure of the English language. Word co-occurrence matrices of English words that are mined from large corpora of English text (often derived from the Web itself) can reveal semantic relations between words. Typically (such in the HAL system [17]) such systems use a window that slides through the text to determine related words. The word "car" is more likely to appear near the word "engine" than an unrelated word such as "kidney." Semantic "neighborhoods" can be constructed in this manner, and then each of these neighborhoods can be used as a method of "keying" into a large set of Web pages. This allows the user to map the semantic space of English onto the semantic space of the Web, and to discover connections between Web pages that are not revealed by the actual links between the pages. A related technique is Latent Semantic Indexing (LSI), which I describe in more detail in Section 2.9.2.

   An alternative way to map the semantic spaces in English to semantic spaces on the Web is to use manually constructed taxonomies of English semantics.

The best known of such taxonomies is WordNet [35], which was developed at Princeton, and consists of a large dictionary of English along with semantic relations between the words in that dictionary. These semantic relations themselves can be used to construct semantic neighborhoods of words which can then be mapped onto corresponding sets of Web pages.

7. Both of the above approaches—using semantic neighborhoods—help reduce cross-talk between unrelated query results. For instance, typing the query "jaguar" into a search engine results in a set of hits. Some of these are about the wild cat, some are about the car, some are about the (Jacksonville, Florida) football team, some are about an old Atari video name called Jaguar, and some are about some chemistry software called Jaguar. Most search engines do not do a good job of separating out these hits from one another (the Northern Light (at www.northernlight.com) search engine is one of the only engines that actually tries to do so[4]). Partly this is due to the fact that it is difficult to separate these results on the fly. I argue that it is necessary to separate these results from one another explicitly during the process of building the index that the search engine uses (that is, during the spidering and indexing process) rather than during the actual search process, because it would usually be too slow to do it during the search and still give very speedy results. Approaches involving the semantic neighborhoods described above can have this effect. In addition, clusters of Web pages can be constructed using traditional methods from the information retrieval literature which involve the vector space model and dot

[4]However, recently Northern Light has discontinued its public web search function, and newer search engines that do separate results have emerged; see section 6.1.6 for an evaluation of these.

products to measure document similarities, but some of these algorithms can be quite expensive in terms of computer time, and therefore must be used with care.

8. Unsupervised techniques such as topic spotting or key phrase extraction can be used to identify topics using key phrases in documents, which can then be used to organize results [37, 124].

One of the main arguments of this thesis is that many of the above activities for finding and organizing information on the Web can be generalized to a single activity which I, following others including [81], simply call *information filtering*, and this is best understood as an activity that is explicitly *social* in nature, rather than individual. This is not surprising, because human language itself, the basis of much of the Web, is a social rather than an individual phenomenon. I use the term information filtering very broadly; I argue that collaborative filtering systems, text clustering systems, manual and automatic text classification systems, and Web search engines are all engaged in information filtering, and the best systems would combine all of the powers of these techniques.

I argue also that information filtering is simply one example of commodity, institutional and social filtering that goes on broadly in society, when people select their cars, their colleges, their mates, their restaurants, and their jobs, for instance. But with respect to the domain of information, which is critical to all of these other activities, since information is used to represent all of these other entities, the reason for the emphasis on information filtering is because individuals and groups have as one of their primary problems in dealing with information, especially in the Internet context, the problem of a signal-to-noise ratios that is lower than desirable. In familiar terms, there is just too much spam and other low-quality material out there.

Web search queries are likely to be governed by Zipf's law [5], which governs such phenomena as the sales of books, box office revenues of movies, the frequencies of words in texts, etc. Zipf's law says that the popularity of the *nth* most popular (or frequently used or consumed) item is proportional to $1/(n^k)$, where k is an exponent close to one. This popularity could be measured in terms of the number of in-links, or in the number of times the page is visited in a given period of time. This is a hyperbolic distribution [72]. Because of this distribution, the lion's share of queries are likely to be taken up by the most frequent queries, so it is these that merit the most attention by those attempting to organize the output of search results.

Too often, as well, the information searching activity is thought of narrowly, in the information science paradigm, of the individual searcher forming queries to a traditional search engine. I think the reason we think about this activity this way is because we have a culture that is very individualistic. I believe that we need to re-conceptualize searching activity as the activity of communities of people who have common interests. Each individual participates in a set of communities, based on her interests.

1.2 Web User Modeling: Interests, Agents, and Feedback

Most Web users have a set of particular interests, and their Web searches are intended to serve these interests. Thus, they are most interested in highly informative sites that are relevant to these particular interests. Each Web search can be viewed as an expression of these interests. If a search engine registers users, and remembers their searches, it can create a profile for each user. It can then use this profile to act as a robot or agent that continually searches out information of interest to that user. People are more likely to invest energy in evaluating information that is within their set of interests. This is because they have a sense of participation in a community, and know that their evaluations will be of use to the community.

Each user's interest makes him or her implicitly a member of a community. For instance, if I like Eric Clapton, I am a member of the community of Eric Clapton fans. This community has a number of Web sites, which are linked to one another.

The Web is already organized into thousands of communities around these particular interests. Some of these communities are explicitly organized as such. For instance, Geocities (part of Yahoo! at geocities.yahoo.com) organizes its user pages in thematic areas, such as computers, politics, entertainment, etc. Webrings (see webring.org) have been created to organize related pages on specific topics into circular linked lists. However, there has been relatively little effort expended to rate pages within specific topic areas and provide users with the pages that are top-rated and contain high amounts of content.

Collaborative filtering, in which users rate items, and then are given access to the aggregated preferences of those with similar preferences to their own, has shown promise in a number of areas. MovieLens (at movielens.umn.edu) has applied collaborative filtering to movies, and Jester has applied it to jokes (at shadow.ieor.berkeley.edu/humor/). Some companies are applying it to consumer product selection. Collaborative filtering goes a step beyond simple polling, such as is used by the Zagat restaurant guides (at www.zagat.com), and the Internet Movie Database (at www.imdb.com).

Most people who use the Web find information either through the use of search engines, which tend to be simple in design, manually organized directories such as Yahoo!, and manually compiled reports such as the Internet Scout (at scout.cs.wisc.edu). Automated user agents that search out information of interest to the user, inform her of it over electronic mail, and then elicit her feedback about it, are not widely deployed, which is disappointing, given all the hype about user agents over the years. One way that such an agent might work is by having a user fill out an online questionnaire with his or her interests. The categories in an existing Web directory (such as dmoz.org) could be used as the database of interests. The interests of each user could be compared with the user-interest database and the user could be informed (via electronic mail) about other categories that they may be interested in (based on the interests of users that have a similar interest set to their own). This is a version of collaborative filtering, where the items being filtered are interests in subject categories.

Users would have continuous access to their set of interests and could update them at will. At a desired frequency (daily, weekly, etc.) the user would receive an electronic mail message with a set of Web sites in it which match their interests and have a high content level. This content level could be measured by the ratings of other users and the number of in-links that each site gets. The email would contain URLs that take the user to a screen with a small window at the top that allows them to rate that site on quality and on how well it matches their interests.

This provides data that can be used to update overall ratings and determine which sites are selected for other users. In addition, users can select sites as being particularly informative, and these selections are then electronically mailed to a sample of users that share similar interests.

Another way to recruit users with specific interests is to post messages to topical newsgroups. One could post a message in the mountain biking newsgroup offering to track activity on the Web related to mountain biking. Users would sign up for the tracking service on a Web page. When sufficient salient links were gathered about mountain biking, each user that signed up for the tracking service would receive an electronic mail containing these links, which they would then be invited to rate.

The salience of page for a particular user would be determined by four factors: the ratings of the page by other users similar to that user (collaborative filtering), the content level of the site (based on some property of the text like the number of category-specific words used, weighted by how specific they are), the number of other pages on the Web that link to the page in question, and the number of other pages the site links to. One may want to discount many links to or from a particular domain, to avoid the problem of web site owners who add extra links just to increase their pages' ranks in search engine results.

1.3 Overview of the Thesis

In large part, this thesis proposes and evaluates novel mechanisms whereby groups of users can work together to achieve effective information filtering. These mechanisms represent some of the steps that I think that would be needed in order to build collaborative information filtering systems like those I have been describing above.

This is done in particular by the use of using link and textual information to classify pages on a topic as outlined in Section 3.3, by merging Web directories and search engines as described in Chapter 4, by a web feedback system as described in Chapter 5, by clustering Web pages as described in Chapter 6, and through a collaboration system as described in Section 7.3. Despite the supposedly individualistic nature of American society, I argue that the logical way to build a Web that serves people is to build a Web that serves groups, and each person gets his or her service from the Web by virtue of his or her membership in several groups. The mechanisms proposed and evaluated herein largely have that as their goal.

Following is an overview of the remaining chapters in this thesis.

Chapter 2 reviews the literature that is relevant to this thesis, including research on Web topology and growth, on page ranking in search engines, on verbal ontologies such as WordNet, on document and concept similarity, and on document clustering and classification.

Chapter 3 explores various ways in which link information might be used in order to improve document classification. First, I experiment with a set of pre-classified pages (all about Wisconsin public-policy topics) and see to what extent link information between these pages can be used to classify them. I also test such a classification with another set of pre-classified pages, which are drawn from a section of the Open Directory (at www.dmoz.org). Finally, I compare a number of variants of the Naive Bayes algorithm, some of which incorporate link information, to see whether the use of link information can effectively improve this algorithm.

Chapter 4 explores the fusion of search engines and web directories. Here, one of the main ideas is to spider off existing pre-classified pages in an attempt to find additional pages which fit in that category or nearby categories. I discuss my efforts to build so-called "vertical portals;" that is, pages on the web that gather information on particular topics using such a strategy. Ultimately, a large set of vertical portals can be merged to create a directory that grows automatically or with the aid of a collaborating community that uses each portal. Such a directory can be merged with a search engine, as well.

Also in Chapter 4, I experiment with a multi-resolution version of the Naive Bayes algorithm which classifies pages first at a low level of resolution (into a broad category) and then at an higher level (into a narrow category). I experiment with this algorithm using both pre-classified pages and pages that are linked off of pre-classified pages. Finally, I see to what extent keywords that are characteristic of a particular category (they appear more frequently or less frequently in that category than average) can be used, in conjunction with inverted-keyword-index search engines, to find more pages that fall into a particular category.

Chapter 5 examines the use of relevance and quality judgments of Web pages drawn from sets of human subjects to rank such pages. Here, the subjects were told that relevance is defined with respect to a category that the subjects are told to judge its extent of membership in, and quality is a global quantity. I gathered a set of such measures by having a group of students rate a set of pre-classified pages. I explore whether the presence of any particular keyword on a page was related to that page getting a high relevance or quality judgment. I also explore whether either of these measures is well-correlated with the page rankings given by public search engines.

Chapter 6 explores some ideas for clustering Web pages. This chapter makes heavy use of lists of pages that are output from ambiguous search engine queries. First, I look at using connected components of the Web (sets of pages connected by hyper-links, that is) to find semantically-related pages. However, I find, by experimentation, that semantically-unrelated pages tend to be connected through very-highly-referenced pages on the Web (such as the Yahoo! home page at www.yahoo.com). Therefore, I prune such pages from the graph, which improves performance, and actually provides quite neat semantic separation without the use of any of the textual information on the pages.

In the next section of Chapter 6, I look at the use of WordNet for disambiguating sets of pages. Here, I again have a set of pages output from a search engine as the result of an ambiguous query, and I experiment to see how well WordNet can separate the pages into separate categories.

In the final section of Chapter 6, I look at the use of HAL-based semantic networks to disambiguate sets of pages. Such networks allow us to see what pairs and sets of keywords occur together. Such pairs or sets can be then used to separate the set of pages that is the result of an ambiguous query into subsets representing each meaning. I suggest the use of a interactive system for doing this, because the machine alone may not function as well as one that combines the strengths of man and machine.

Chapter 7 discusses various systems that have been built in recent years for collaborative filtering, and closely related discussion systems which attempt to keep track of the prestige of the various participants in the system. I suggest that such systems can be merged with vertical portals and search engines. I also build a simulation of a system in which both documents and authors get ratings, and the authorship relation between documents and authors is preserved, and propose embedding such a system in the Web bymaking use of meta-data about authorship.

In Chapter 8, I conclude with a summary of the results of the thesis. I also propose in this chapter an integrated system which incorporates many of the techniques and ideas I have proposed in this thesis.

Chapter 2

Background

This chapter reviews literature that is relevant to various parts of this thesis. This literature includes research on Web topology and growth, on the relations between the Web and real-world entities, on the electronic dictionary system WordNet (which I use in Chapter 6), on how to rank pages in Web search engines, on document and concept similarity, on document clustering and related techniques, and on how to classify and cluster web pages. The reader familiar with the any or all of the material covered in this chapter may feel free to skip some sections or the whole chapter. All the background material I am presenting for this thesis is found in this chapter, with the following exception: the background to the collaborative model built in Section 7.3 is found in Section 7.2.

2.1 Web Topology and Growth

I include this section to give the reader a sense of the problems of scale that are faced by architects of search engines and directories; the task of information management on the Web is colossal, and growing daily. This thesis is largely concerned with such information management, from the point of view of the individual user and of groups of users. I also include this section to give the reader a sense of the topology of the Web, which will come into play in this thesis especially in Section 6.1, which is about the use of connected components for semantic disambiguation.

Mathematically, the Web can be thought of as a directed graph. Because the Web is a social artifact, the result of the collective activity of a large number of people, who are acting somewhat independently, we might expect that the connectivity structure of the Web would be non-random, but not regular. It turns out, upon empirical investigation, that the Web is an instance of a "small world" network. Small world networks are networks that "can be highly clustered, like regular networks, yet have small characteristic path lengths, like random graphs [121]." This is not surprising, because the Web reflects social networks, and small world networks are common in the social world, and are illustrated by the parlor game "Six Degrees of Kevin Bacon."

A player in this game is given the name of an actor and is required to connect that actor with Kevin Bacon by a chain of no more than six movies. This is usually possible, because Kevin Bacon has been in a lot of movies and (more importantly) the social structure expressed by movie casts has a low mean distance between actors.

It is difficult to study the topology of the Web in full because of its size (Google says that they index slightly under 2.5 billion documents in mid-2002, and this is growing rapidly); even the most exhaustive search engine (at this writing, supposedly Northern Light, but this is disputed by Google and others) reportedly only covers somewhat over a third of the pages on the Web [66].

By studying the local structure of a very large Web site (wd.edu, at the University of Notre Dame), and then extending its properties to a larger model, Albert, Jeong, and Barabasi [3] have calculated that the present diameter of the Web is about 19 links, meaning that, on average, it takes no more than 19 links to get from any random page on the Web to any other page. (Of course, finding this path is a computationally intensive task in practice, and in some cases, pages are not connected at all). In addition, this path length is expected to grow only with the log of the number of pages. This is because, for instance, you can increase the path length by only one by adding ten new pages linked off of each existing page, while increasing the number of pages by a factor of ten.

Thus, exponential growth (if it exists, and if it continues) in the size of the Web will only increase this value (of 19) by a few more links, if that, in the next few years. Albert et al.'s work fits in well with the work of the Clever project [58] on "authority" and "hub" sites; it is likely that these sites pay a major role in connecting relatively obscure pages. For instance, a page on jaguars (the animal) (an obscure one not listed in Yahoo!) could link to a page that was authoritative on jaguars (say, one put out by the Bronx Zoo). The latter page is listed in Yahoo!, and Yahoo! then links this original page to other obscure pages through other authoritative pages that it links. Yahoo! is very large in size, the Bronx Zoo's site is intermediate, and the obscure sites consist of a single or a few pages. The hierarchy of the Web reflects the hierarchy in the social world.

Albert, Jeong, and Barabasi also found that the number of in-links and out-links on any particular page is approximately governed by a power law. The probability that a given page has 10^k links inward is given by about $k - 2.45$ for out-links and $k - 2.1$ for in-links. So there are very few pages with very high numbers of in-links and out-links; most pages have a low to moderate number. Huberman and Adamic [53] found a similar power law in the size structure of the Web. They found that the probability that a site (defined by a root domain name, such as www.wisc.edu) has 10^k pages is approximately proportional to $k - b$, where b is a constant. Thus log probability versus log size appear as a straight line. They got this information from two Web spiders, which regularly "crawl" Web sites looking for pages. This result allows one to estimate what fraction of a particular set of sites will have a given size.

Broder et al. [15] verified the power law for Web in-links on a larger Web crawl than that done by Albert, Jeong and Barabasi. They also discovered something about the topology of the Web. If viewed as an undirected graph (reducing directed links to undirected ones), they found that over 90 percent of the pages in their 203-million-page crawl form a single connected component, and the rest of the pages are not connected to this large component. If one looks at this large connected component from the point of view of directed links, it is divided into four pieces, which they call SCC, IN, OUT, and TENDRILS.

The first is a strongly connected component (SCC) which contains about 56 million pages. (A strongly connected directed graph is defined as a graph in which any node can be reached from any other along a path consisting of directed links.) The IN piece contains pages that can reach the SCC (again, along directed links), but cannot be reached from it. The OUT piece contains pages that can be reached from the

SCC, but cannot reach it. Finally, the TENDRILS contain pages that cannot be reached from the SCC and cannot reach it. (Note that each of these last three pieces is not a strongly connected or connected graph). They find that the central core has a diameter of at least 28, and that the overall diameter of the graph is over 500. They also found that the chance that any path (directed or undirected) exists between any two nodes chosen at random is only 24 percent, and that if a directed path exists, its average length will be about 24. If an undirected path exists, its average length will be about 6.

2.2 The Web and the Real World

To a large extent, the Web reflects the structure of the actual world. Not only do pages, like traditional texts, refer to and describe objects, events, people, places, and institutions, but the Web reflects part of the actual structure of the (social) world. For instance, companies and non-profit institutions have home pages which reflect some of the properties of those entities. Many people have home pages as well. The pages of these institutions and of the people tend to be related to one another that is similar in structure to actual social relations. So, for instance, one institution tends to have links to other institutions of which it is a part or with which it interacts. Of course, the Web is still elitist; only the more wired among us have much of a presence on the Web at all, although many—perhaps most—people are at least listed in online phone directories (at least in developed countries). Projects like Lenat's CYC [67] (built by his company Cycorp, at www.cyc.com) have attempted to construct an ontology in symbols (typically in predicate-logic formalisms such as can be processed by languages such as Prolog and Lisp, as well as more ordinary languages like C and C++) that reflects the actual ontology, the hierarchical and compositional structure of entities, objects, actions, and events, that exists in the real world. Tim Berners-Lee, the founder of the Web, has more recently been promoting the idea of a "Semantic Web" [10].

This is a version of the Web in which Web pages contain not only the direct HTML or XML [12] that allows them to be rendered by a browser, but also meta-data description about the content of the Web pages that is readable by other computers. Some of this meta-data may be found by information extraction (e.g. [116, 19, 24]), and some may be explicitly provided by the Web page authors.

So, for instance, a page for a doctor's office would not only contain whatever ordinary readable information, such as a description of the doctor's credentials, the office location, the hours, etc., but these data would also be encoded in a predicate logic form so that other computers (clients) on the Web could read and process them. So, for instance, a traveling patient seeking an appointment with a doctor might have her mobile client contact her main doctor's office server, which processes an automatic referral to a trusted network of other doctors, which queries this network to see which doctors are within a particular radius of the patient's current location. Then the schedules of the patient and the schedules of the doctors negotiate a time and an appointment is set up. Medical histories are transferred. All of this needs to be done with proper attention to user authentication and privacy considerations.

Thus the construction of machine-usable meta-data to accompany Web pages is very important to such a vision of a future Web. Classfication, clustering, and group collaboration as described in this thesis can help construct such meta-data.

2.3 WordNet

WordNet [35], an electronic English dictionary and semantic network built at Princeton under the direction of renowned psychologist George Miller, contains an ontology similar to that proposed for the Semantic Web. This ontology is implicit in the hierarchy of concepts that exists within it. WordNet was built to create a dictionary that more accurately reflects the relations between words than traditional alphabetic directories, or even thesauruses, do. The latter lists synonyms, but synonymy is only one of many relations between words, as we will see below. WordNet is useful as an interactive dictionary, and it has had wide-ranging applications in automated processing of natural language, including applications in knowledge extraction (e.g [6]), text categorization (e.g. [43]), document analysis (e.g. [117]), and sense disambiguation (e.g. [71][110]). The WordNet Web site at www.cogsci.princeton.edu/~wn lists many publications that make use of WordNet. I make use of WordNet in Section 6.2, which describes experiments in using WordNet to disambiguate the results returned by a Web search engine.

Since WordNet's hierarchy reflects the actual world, it can be used as a directory for the purposes of classifying Web pages. That is, individual concepts in WordNet can serve as categories for classification of Web pages. However, it is not complete. For instance, while it "knows" about a "university," which has three senses in WordNet, it does not know about the University of Wisconsin, which Yahoo! and other Web directories know about, because their directories have been constructed manually on the basis of large numbers of external pages, whereas WordNet's designers were not interested in encompassing the entire world in their ontology, but only the world that is described by (mainly individual) English words. Thus WordNet lacks a lot a specific knowledge, especially about proper nouns. For instance, WordNet only contains one meaning of “jaguar,” its literal meaning as a type of wild cat, and not the way the word has been used to name other things, notably cars, a football team, and a video game. Since WordNet’s ontology is designed to describe the semantics of words, not of whole documents, WordNet only links nouns with their synonyms, hypernyms, hyponyms, meronyms, and coordinate terms (described below).

It does not link words that are semantically linked by virtue of belonging to the same topic or activity. For instance, “photograph” and “darkroom” are not linked in WordNet, even though they are both part of the activity of photography. In fact, “photograph” and “photography” are not even directly linked in WordNet. Words in WordNet are arranged in synsets, which are sets of words with the same meaning. Thus, each synset in WordNet represents a distinct concept. Concepts are not described explicitly. Rather, they are described in terms of the set of words in their associated synset.

Thus two senses of the world “board” are expressed by two synsets. The first contains the words “board” and “plank” and the second contains the words “board” and “committee.” A noun, if it has many senses (meanings, that is), is listed in multiple synsets. For instance, the word “board” has 13 senses in WordNet, 9 as a noun, and 4 as a verb. It turns out that there is a close relationship between the number of synsets a word participates in and its frequency in the language. The word “board” is a relatively frequent word, and has a lot of senses. WordNet also contains many short phrases to describe concepts that serve as hypernyms of actual words. So, for instance, the phrase “medical care” is in a synset with the phrase “medical aid" and has hyponyms that include "therapy," "nursing care," and "disinfection."

"Therapy" in turn has hyponyms "psychotherapy" and "chemotherapy," among others. A hypernym of a word is a word that describes a super-set of a given word. So, for instance, the word "canine" is a hypernym of the word "dog," and the word carnivore is a hypernym of the word canine. Hypernymy is transitive, so the word "carnivore" is also a hypernym of the word "dog." Hyponymy is the opposite of hypernymy, and is also transitive, so "dog" is a hyponym of "carnivore" and of "canine." Coordinate terms are terms that are sisters in the tree formed by the hypernymy-hyponymy relation. Thus "lion," "tiger," and "jaguar" are all coordinate terms, as all being types of cats. Coordinate terms are clearly useful in determining the "semantic neighborhood" of a document.

One can think of a set of coordinate terms as corresponding to a particular topic. If a document contains a lot of coordinate terms, it is likely to be at least in part about the corresponding topic. So if "lion," "jaguar," and "tiger" all appear in a given document, chances are high that the document is at least at part about cats, although it is possible to think of counter-examples, for instance, the sentence "I drove my Jaguar (a car) to the Food Lion (a supermarket) and on the way my girlfriend kissed me and called me a real tiger (metaphor)." However, in general, this makes a good heuristic. A query of "+jaguar +lion +tiger" in the AltaVista search engine found pages that were mainly about wildcats, although there was also a page that listed sports teams that used those names, a page for mascot costumes based on those animals, and a page where spies used those names as code names.

But the vast majority of the results were pages that were actually about wild cats, which indicates that the presence of a large number of coordinate terms is a good heuristic for assigning a topic to a page. Meronymy captures the part-whole relation in WordNet. The word “thigh” is a meronym of “leg,” and “leg” is a meronym of “human body” in one of its senses, of “chair” or “table” in another, and of “journey” in yet another. Homonymy is the opposite of meronymy; thus “journey” is a homonym of “leg.”

You can see how WordNet could therefore be used for knowledge discovery from texts. For instance, if the word “leg” and “journey” appear in close proximity in a text, a program might conclude from this that the sense of leg being used is the one related to a journey. Of course, it is easy to think of counter-examples, for example, the sentence “the journey was so long that my legs were tired.” WordNet has many applications in linguistic research and natural language processing, not all of which are immediately relevant to our purposes. For instance, WordNet has application to simple word-sense disambiguation (in the context of single sentences), but this is not something that we are immediately interested in in our efforts to classify and organize Web pages.

In the information-retrieval context, and for the Web, WordNet can be applied in a variety of fashions. Web pages can be analyzed in terms of their “semantic neighborhoods.” That is, if a page contains a set of words that are all close to one another in WordNet, this neighborhood can be said to define one of the topics of a page. A given page may be in more than one of these semantic neighborhoods. These neighborhoods can be used to automatically create Yahoo!-style directories out of pages. However, these neighborhoods have their limitations, because words that are not linked by one of the relations that WordNet uses but are actually semantically linked in some other way will not be discovered.

To give a greater specificity to the idea of a semantic neighborhood, consider the following. Consider all the unique words on a particular Web page, after pruning out very high frequency words. Then build all of the graphs that connect these words by WordNet relations. For instance, if a particular page contains the words "lion," "jaguar," "tiger," "heart," "liver," and "stomach" (perhaps it is a page about veterinary medicine of big cats) then the first three of these words are coordinate terms in WordNet, as are the last three words. Each of these sets of three words represents a small graph, which in this case is a completely-connected clique with a triangle shape.

Each of these cliques can be viewed as a semantic neighborhood, and has (perhaps) identified a topic of the page. (Normally, one would want to have a cutoff for the minimum size of a semantic neighborhood, which seems to me would have to be minimally three connected words; if you included all pairs of connected words, the number of semantic neighborhoods associated with each page may be too great. This may be the subject of experimental tuning to find minimal neighborhood sizes that lead to the best performance, in terms of topic identification of pages and in terms of clustering of pages.) The words in each of these semantic neighborhoods can then be (separately, for each neighborhood, or together) passed to a search engine to identify additional pages that are on a similar topic as the page in question. This can be used to add a "find similar pages" facility to a browser or search engine.

Similarity between Web pages that one has already found (say, as the result of a search-engine query) can be gauged by the degree to which they share semantic neighborhoods in WordNet. For instance, if one page contains the words "lion," "jaguar," and "tiger" and another contains the words "cheetah," "lion," and "leopard," these pages have overlapping semantic neighborhoods, which overlap at the node "lion." The number of overlapping neighborhoods, and their degree of overlap, can be used as a measure of the semantic similarity between pages. If you have a set of pages, these pairwise page similarities can be used to cluster pages, in the manner standard in the information-retrieval literature. This is a similar idea to what I have suggested elsewhere; the idea of using a hierarchical supervised learning algorithm (e.g. a hierarchical, multi-resolution version of the Naive Bayes algorithm[1] which I implement in Section 4.2) to separate search engine results by classifying the results into the tree and then displaying the results organized by categories. The difference is that that here the source of semantic information is WordNet, whereas in the other case, it is pre-classified Web pages. Both make use of extensive human effort (WordNet and Dmoz respectively).

WordNet can also be used in data mining and information extraction from Web pages. Typically, such tasks are defined as the transformation of free text material and other free-form data into data that is in a structured (fielded) form suitable for further processing. The Web has been viewed as a major potential source of new structured material, see, for example, [25]; the process of gathering it has been referred to as "text data mining" by Hearst [49]. Hearst [48] has built a system for extracting new instances of WordNet relations (that is, the relations, such as hypernymy and synonymy, described above) directly from text. These relations can be used to extend WordNet, and then the extended version of WordNet can be used in other systems.

Large numbers of random pages drawn from the Web can provide text and create word frequency tables for all the words in WordNet (which has a large subset of the

[1]The ordinary, non-hierarchical version of this algorithm is used widely, for instance, in [55, 63, 87].

words in English, except for proper names and rare words). Such tables are useful in text processing, because the presence of less-frequently used words in a given document tend to characterize that document better than the more-frequently used words in the document. For instance, a document that contains the word "phlebotomy" (surgical insertion into a vein) is likely to be on a medical topic. If this document also contains common words like "will" or "do," this tells us very little about its topic. WordNet allows us to determine that this topic because "phlebotomy" is in the ontological tree below "medical care," one of its hypernyms.

WordNet can also be used in more pedestrian ways. For instance, it can be used to determine if a given word is in fact an English word and what its base form is (although WordNet does not contain many rare words, so this is a drawback). WordNet can discover base forms because it understands the inflectional morphology of English.

2.4 The Clever Project: Hubs and Authorities

The Clever project at the IBM Almaden research laboratory [21, 58] distinguishes between two varieties of important pages on the Web: authorities, which are pages that are pointed to by a lot of other pages, and hubs, which are pages which point to a lot of other pages. Hubs are useful because they centralize a lot of information about other important pages, and authorities are important because they have a lot of sites pointing to them and thus are likely to contain useful information, or at least be popular. The Clever project's approach is good at identifying pages that are likely to be important or useful (although it may not be good at finding pages that were recently added to the Web and therefore do not yet have many in-links). In addition, it may discount pages that are in a relatively less dense area of the Web. For instance, there are many pages that contain the word "jaguar." These pages may be about the car, the Atari video game, the Jacksonville Jaguars football team, or the actual wild cat that all of these other things are named after. Ironically, pages about the animal appear less frequently than its three namesakes, presumably because Web page authors are more interested in the namesakes.

The Clever project uses an iterative algorithm to compute scores for authorities and hubs. The higher an authority's score is, the better an authority it is said to be; likewise for hubs. An authority's score is determined by summing up the hub scores of hubs pointing to it. This is done for all authorities, and then the scores of all hubs are updated by summing the authority scores of all the authorities that each hub points to update that hub's scores. Then the scores are normalized by requiring that the sum of the squares of each set of scores sum to one. These steps are repeated until the scores converge, which they find (experimentally) occurs after about 20 iterations. Note that this algorithm has complexity that is related to the number of pages being processed and the density of links between them.

Kleinberg proves [58] that this algorithm is equivalent to finding the principal eigenvectors of the matrices A^TA and AA^T, where A is the adjacency matrix formed by viewing the connections between pages as a graph. These two principal eigenvectors contain the hub scores and the authority scores. Since there are well-known methods for computing the eigenvectors of a matrix, it is not necessary to use the iterative algorithm, but they retain it for its explanatory value. Kleinberg finds that the non-principal eigenvectors sometimes contain useful information as well; for instance, on the query "jaguar*," the pages having the highest values on the principal eigenvector were about the Atari Jaguar product (a video game), the pages having the highest positive values on the 2nd non-principal eigenvector were about the Jacksonville Jaguars football team, and the pages having the highest positive values on the 3rd non-principal eigenvector were about the Jaguar automobile. The cliques or clusters that I derive in Section 6.1.3 appear to be akin to these components, although they are not derived in the same manner, but rather one that is more intuitive; see that section for more details.

On the query "abortion," the 3nd non-principal eigenvector separated pro-choice sites (which had positive scores in the eigenvector) from pro-life sites (which had negative scores). However, this is not a reliable way to find topics, even though Kleinberg points out that nodes (in a graph) having relatively high positive values on a particular non-principal eigenvector are often well-separated from nodes having relatively high (in absolute value) negative values, and therefore may be on different topics.

In general, one would expect that pages on a given topic will be on average further away from pages on a different topic than they are from pages on the same topic. Of course, this is only true on average, since pages on completely different topics may be linked closely. For instance, if I like chimpanzees and Eric Clapton, pages on each of these topics may be linked from my home page. Note the hub and authority scores of a particular page say nothing about the content of a particular page, simply that it is likely to be useful. They do not tell you anything about what the topic(s) of the page are, or how much information the page contains on a topic. In fact, many hub and authority pages may not contain much actual information, but are jumping off points, such as Yahoo!, or a corporate or institutional home page. Some pages that contain a good deal of information, on the other hand, may not be linked into by many other pages.

2.5 Page Ranking Techniques

Many of the major search engines do not disclose the details of their operations, but it is widely believed that most search engines have traditionally used two factors as the main considerations for determining how to rank their search results. The first of these factors is the presence of the search keywords. The more frequently the search keywords appear, and the more of them that appear on the page, the higher the page is ranked. The other factor is the number of other pages that the search engine knows about that point to a particular page. The pages with the higher in-links can be ranked higher. Of course, both of these techniques lend themselves to a game of cat and mouse between those maintaining and promoting Web pages and those running search engines, because there are things that Web page maintainers can do to raise both of these values, such as putting common search words in the META tags of their HTML source or creating a lot of dummy Web pages that do nothing but point to pages that they want to drive traffic to, by artificially boosting their rank in the results of search engine queries.

The Google [14] search engine has been widely recognized as having provided a partial solution to these problems. Google's PageRank algorithm effectively measures the overall popularity of a page in the global Web. All the pages that Google "knows" about are assigned a positive-valued PageRank, and all the PageRanks sum to one.

Thus, in technical terms, the distribution of PageRanks is a probability distribution[2]. The PageRank of a particular page represents the probability that a Web surfer, starting at any random place in the graph of the Web, will reach that page next. Thus, the pages with higher PageRanks have more paths leading to them.

Formally, Google's designers define the PageRank of a particular page u as the following:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v} + cE(u)$$

where $R(u)$ is the rank of page u, $B(u)$ is the set of pages pointing to u, N_v is the number of pages v points to, and c is a normalization constant inserted so that all the ranks add to one. $E(u)$ is a vector added to adjust for the problems of "rank sinks," for instance a pair of pages that point to each other and have one other page pointing to one of them. Without this $E(u)$ factor, such pages could accumulate rank, draining it from other pages. The authors show that if R is initialized to practically any value, then the PageRank will converge to the same final value, reflecting each page's accumulated weight from all the pages pointing to it, either directly or via a chain (because the iterative nature of the algorithm allows the information to propagate).

I argue in this thesis that one needs to go beyond global ranking of pages; instead each page needs to be ranked in a manner specific to the topic in which it appears, after they have been separated into topics. I discuss how I have done such a contextualized rating for a sample system in Section 4.1.

[2]This is because a probability distribution is defined as a distribution of positive numbers between zero and one that sum to one.

2.6 Spreading Activation

Another popular way of thinking about using links between pages to facilitate Web use is by using the old concept (familiar to students of both artificial neural networks and semantic networks) of spreading activation (see, for example, [4]). Pirolli, Pitkow, and Rao [90] use this concept, and the concept of information foraging, in their efforts to extract usable structures from the Web. Information foraging is the activity of a user looking for information in some Web locality, which is some local part of the directed graph that is the Web. It is important to think of Web searching activity as part of the overall cognitive search activity of the learning human organism, which Belew refers to as "finding out about" in his book on Web searching [9].

The concept of information foraging comes from the idea of an animal foraging for food in an environment. Since information is a valuable resource, as is food, and time, the goal of a foraging activity is to get the best food (or information) in the least amount of time. The quality of information that is foraged from the Web is measured by the information gain, a familiar concept from information theory [111]. Users of the Web, and of search engines and Web directories, can be thought of as information foragers. One way that users can improve their foraging activity is to identify ways to categorize information that allows them to quickly identify pages that are likely to give them valuable information.

In addition, the idea of foraging fits in nicely with the idea of creating better visualizations of the Web that allow people to more rapidly pull out what interests them and what is likely to be most informative. Pirolli et al. discuss several ways in which users can categorize pages. Placement of Web pages into sets can be useful in navigation, but pages on the Web are not explicitly placed into categories or classified into types by their authors (except in some cases through the use of the "META" HTML tag, which allows page authors to publish meta-information about pages, such as keywords with which a page is associated).Thus it is necessary to induce the type of a particular page.

Their taxonomy of types of pages follows. First, there are home pages, which they call head pages, which may be personal or organizational. Second, there are index pages, which are similar to what the Clever project calls hub pages. Third, there are home pages of sites that have been explicitly built around a particular topic to serve a community, which may not have a formal organization. A reference page is a page that is used as a reference on a particular concept. A destination page is a kind of reference page that other pages point to but does not point to any pages. Finally, a content page contains information and is not primarily for navigation.

Pirolli et al. note that there are several kinds of information that are available about a page. Its topic (via text similarity) and its connectivity to other pages (its topology) are familiar to us. They note, however, that the URL, the file size, modification date, and the page's usage statistics may also be useful in categorizing the page. The URL may indicate where the page belongs in the semantic hierarchy of its site.[3] Note that the usage statistics are typically only available to those running the Web server and to the Webmaster of a particular site, not to the general public or to search engines. The following statistics were kept by Pirolli et al. about each page: its size, number of pages pointing to it ("in-links"), number of pages it points to ("out-links"), the number of times it was requested (this information is only typically

[3]Often, as has been pointed out in [45] and by others, the structure of a Web site better reflects the structure of the bureaucracy that built it rather than any other conceptual structure.

available in the server log of the Web server, but they had access to this for the site with which they were working, the Xerox site), the number of times the page was the first page visited on a user's traversal, and the textual similarity between it and its children (pages to which it pointed).

They used these statistics to categorize pages based on the taxonomy that they set up. They created a linear set of equations based on this set of features, where the output is the type of page (there were 8 types). They set the weights used in the set of equations manually. For instance, they assumed that content nodes would have few in-links and few out-links, but relatively high file sizes, and thereby selected weights of -1, -1, and +1 respectively on these features. They solve these equations, and report precisions for the automatic assignment of pages into the manually-assigned categories ranging from 0.30 to 0.99, with all but one above 0.50.

The authors argue that the attachment of more meta-information (in programming language lingo, assigning types; in set theory lingo, assigning set membership) to pages can be a boon for user navigation and visualization of the Web. They give examples like the following. If a Web surfer wants to find personal home pages that are most similar to a particular personal home page, the system can use spreading activation from the initial home page, stopping at a fixed number of steps, and output all the other pages visited along the way that have also been identified, by mining meta-information, as personal home pages. This is akin to the directed spidering that I use in Sections 4.1 and 4.2, in which I start from pre-classified pages and use a classification algorithm to determine the class membership of pages linked off of those pages.

2.7 The Term-Vector Model of Document Similarity

The term-vector model in text classification and clustering considers each document as a vector of the terms appearing in it [5]. Such a model is widely used to create structural relations within sets of text documents [105]. The terms (i.e., words) in each document may be represented in a binary fashion (if a term is present it is coded as 1, and if it is not present, as 0) or they may be coded by the numerical frequency of a term in a document. Using such a representation, one common way of estimating the similarity between documents is measured with the normalized dot product (cosine) of the term vectors representing each document. Of course, there may be hundreds of terms in each document, so these term vectors can be very long. I make use of the term-vector model in my experiments with WordNet in Section 6.2. Term vectors can also be normalized by dividing each value in the vector by the sum of the squares of all values.

It is difficult to estimate the semantic similarity of documents because two documents can share no words in common yet be about the same topic. This is because there are typically several synonyms that express the same underlying concept. Attempts to get around this problem typically use thesauruses, which list synonyms. WordNet contains such a thesaurus among its components. Of course, the use of thesauruses or other semantic information about terms in a document tends to increase the number of terms describing a document and therefore increase the cost of computing similarity.

2.8 Concept Similarity

Ultimately, document similarity is based on the similarity of individual words and concepts in each document. Jiang and Conrath [54] discuss ways to compute lexical and concept similarity. One way uses the idea of the information content of a concept in order to compute word and concept similarity. This is called an information content-based or node-based approach. In information theory [111], the information content of a concept is defined as the reciprocal of the log of the probability of observing an instance of that concept (among all possible observations of all concepts). Since the probability of observing an instance is lower the more specific a concept is, the information content is higher. This fits well with an intuitive understanding of what "information content" means; telling me that something is an object does not tell me much about it, while telling me that it is a jet airplane tells me a good deal more. Telling me that it is a vehicle is intermediate in content.

Thus, if you have a ontology of concepts organized in a tree—which WordNet also contains, along with the thesaurus information—the specificity and information content of concepts increases monotonically as you move down toward the leaves of the tree. Note that the information content of a document could be defined as the sum of the information content of the distinct concepts that appear in that document. This could be a way of distinguishing between documents that contain very little information and those that contain a good deal, which could be useful on the Web, because the Web contains many documents that contain very little usable information.

One way to define the similarity of two concepts is by the information content of the most specific concept that subsumes both of the concepts. To determine the similarity of words that each have multiple senses, one uses the maximal similarity of all the possible concept pairs. It still remains to estimate the values of the probability of an instance of a concept. Jiang and Conrath [54] discuss several ways of doing this. One way is simply to define the frequency of a concept by the total frequency of all the words signifying it, which are all the words underneath it in the ontology tree. This does not account for the fact that some words may have many meanings; one way to try to correct for this is to assume that the word's occurrences are evenly divided among the concepts that it signifies, and therefore assign only an equal fraction of each word's occurrences to each concept. Whichever way the frequency of a concept is defined, the probability of a concept is then estimated by its frequency divided by the total frequency of all concepts.

Another way to compute concept similarity is what Jiang and Conrath refer to as the edge-based method. Concept similarity increases as the distance between the concepts in a semantic network such as WordNet is reduced. However, there is no reason to assume that the weights assigned to each edge should be equal. For instance, it has been argued [97] that if particular parts of a semantic network are particularly dense, the distances between nodes should be reduced. Also, Jiang and Conrath argue that distances should shrink as one moves down the tree toward the leaves, since sister nodes are more similar as one moves down. In addition, if the semantic network contains different kinds of links, such as IS-A, PART-OF, and SUBSTANCE-OF links, one may want to assign different weights to these.

2.9 Document Clustering

Traditional document clustering methods in the field of information retrieval are of at least three types. The first type of method, which Salton and Wong [106] refer to as a hierarchical grouping method, constructs a complete n by n similarity matrix for the n documents to be clustered, and then uses this matrix to construct the categories, by grouping together first the most similar documents. This is of complexity n^2, at a minimum, and is impractical for large collections of documents, such as we find on the Web. In the second type of method, which Salton and Wong refer to as iterative partitioning methods, a fixed number of clusters is set up initially, and then the remaining documents are assigned in turn to the cluster whose centroid to which each document is most similar. The centroid represents the "average document" of a set of documents (here, a cluster); that is, it is the average of all the term vectors in the set. Unlike a real document, the centroid can (and virtually always does) have fractional values for some of terms in the term vector representing it.

Chapter 6 is devoted to clustering Web pages, and presents three different techniques; see the chapter for details.

The initial clusters may be the result of manual classification (in the context of the Web, categories that are selected by a directory such as Yahoo! may be used, or sites that are highly referenced by hyper-links from other sites may be used with the idea that each of these reflects a particular topic), or regions of high density in the term-document space may be used. The number of clusters may be permitted to change during the course of the algorithm, by merging or splitting clusters. A hybrid method may be constructed by using a random sample of documents and clustering them according to the more expensive hierarchical grouping method, and then adding the remaining documents to the clusters so found by an iterative method.

In the third type of algorithm, which Salton and Wong refer to as a single-pass method, because it only involves a single pass through the data, a tree is used in which clusters are grouped together as one ascends to the root. The algorithm considers each document in turn and use the tree to place the document in the appropriate cluster (the one it is most similar up to that point). Such algorithms are of complexity $n \log(n)$, since the tree height is proportional to $log(n)$, and typically the height of the tree is navigated in placing the document in the appropriate cluster. The tree is constructed by first constructing the root node and then creating daughter nodes as each node gets too large and needs to be split. A review of hierarchical methods for document clustering can be found in Willett [125].

We review some methods for clustering that have been developed since Salton and Wong's early (1978) review in the next few sections.

2.9.1 Suffix Tree Phrase Clustering

Document clustering can be useful in organizing the often chaotic results of search engines. However, as Zamir and Etzioni [130] point out, if document clustering is to be used in a practical search engine, the algorithm must work quickly, because users expect fast results and the CPU cycles that can be devoted to any particular query are limited, since typical search engines receive millions of queries a day. (The alternative is to cluster all the pages that the search engine knows about continuously, and then present all the outputs of a query in a clustered fashion, but even in that case, performance is a serious issue, given the large number of documents to be clustered.) The best document clustering algorithms are $O(N)$ where N is the number of documents returned by the query. $O(N \log(N))$ algorithms are likely to be acceptable, but $O(N^2)$ or worse algorithms will not be, given the large number of documents often returned by a query.

Zamir and Etzioni note the following requirements for a clustering algorithm: it should separate documents into those relevant to a user's query and those that are not, it should produce summaries of each cluster which convey to the user what that cluster is about, it should permit the assignment of pages to more than one cluster (because pages have multiple topics), it should cluster based only on the snippets of the documents returned by the search engine (because users are unwilling to wait for the search engine to get the original documents off the Web), it should be fast, and it should be incremental, processing each snippet as it is received. To meet these requirements, they devised an algorithm, which they called Suffix Tree Clustering (STC), which is $O(N)$.

STC is based on identifying common phrases in documents, and then clustering together those documents that contain common phrases. Since it uses phrases, not individual words, for clustering, it differs from most traditional clustering algorithms that use a document term vector based on individual words that pays no attention to the relative location of the words in the document. STC is based on the idea that phrases will do a better job in clustering documents than single words or combinations of words. The idea is plausible.

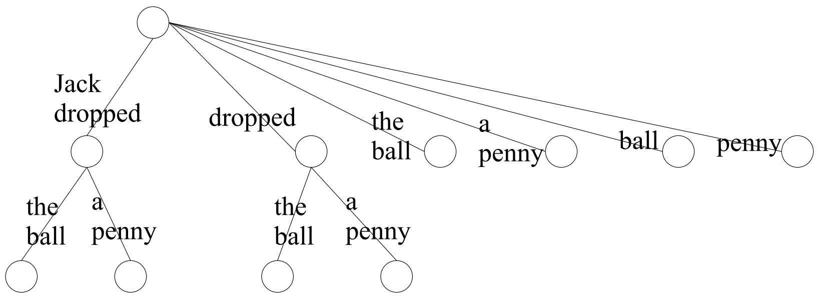
To introduce the concept of a suffix tree, I define "suffix" and "trie." A suffix of a sequence of words is a sub-sequence of those words up to and including the last word. The null sequence is not a suffix, in the definition used here. So, for instance, the suffixes of the phrase "jack dropped the ball" are the whole phrase itself, "dropped the ball," "the ball," and "ball." In the suffix tree, these suffixes are represented by paths from root to leaf. A trie is a type of tree in which the nodes are empty and the data is stored on the edges between nodes; it has been used to implement algorithms such as Huffman coding [59], and frequently in information retrieval. Tries are compact representations that compress data.

A suffix tree [123] is a rooted, directed tree; specifically, it is a compact trie containing all the suffixes of a particular sequence of words. Each internal node has 2 or more children, and every edge is labeled with a sub-sequence of a given sequence of words S. They define the "label" of a node as the concatenation of the edge-labels of each edge from the root to that node. No two edges toward the children of an internal node in a suffix tree can have edge-labels that begin with same word. For each suffix of S, there exists a node, which they call the suffix-node, whose label is that suffix. (They use each sentence of each document in their collection of Web pages for their sequences S.) Note that suffixes are of lengths up to one less than the length of each sentence.

A suffix tree for the phrases "jack dropped the ball" and "jack dropped a penny" is shown in Figure 2.1.

A base cluster is defined as a set of documents containing a common phrase. The leaf nodes of the trie represent phrases and are labeled with their corresponding base clusters. Zamir and Etzioni assign scores to base clusters. The score of a given cluster is a function of the number of documents in that cluster and the length of the associated phrase, not including words on the stop list. Phrases of length 1 are penalized; there is a linear relation between the length and score for phrases of length

Figure 2.1: A Suffix Tree for the Phrases "Jack dropped the ball" and "Jack dropped a penny"



2-6, and beyond that, a constant multiplier is used.

Zamir and Etzioni note that there may be a large degree of overlap in phrases between documents. They deal with this by merging base clusters that contain similar sets of documents. They use a binary similarity measure, where two base clusters are considered to be similar if half of the phrases that are in one are also in the other. If two base clusters are similar, they are merged. This creates a new set of base clusters, with multiple phrases associated with each base cluster.

In practice, this is used to do an on-the-fly clustering of search engine results. They build these base clusters for a particular set of documents returned. They select the base clusters containing the largest numbers of documents. For each of these, they output the set of associated documents along with the set of associated phrases.

They attempt to compare their algorithm with other clustering algorithms. Because clustering algorithms are unsupervised, there is not really a good way to do this. They attempt to overcome this by setting up a set of categories manually for a set of pages and see how well a variety of algorithms do on placing documents correctly (as defined by their own manual classification of pages in this set) in these categories (clusters) (i.e., a precision metric). They find that their algorithm outperforms a wide variety of standard clustering algorithms. These include agglomerative hierarchical clustering [120], the K-means method [99], the single pass method [51], "Buckshot" [26], and "Fractionation" [26]. Of course, they are subject to the criticism that their evaluation method may be far from unbiased.

2.9.2 Neural Network Methods for Text Clustering

Neural network methods [50] can also be used for clustering, but because of the number of training epochs that are needed to for a neural network to converge to a stable value, are considerably more computationally expensive than the methods given above. For instance, if one has n documents that one plans to cluster into $log(n)$ classes, and there are f features per document, a typical neural network architecture connects a unit representing each of the f features to a unit representing each of the $log(n)$ clusters. A single weight-update pass through the data therefore involves a minimum of $nf \log(n)$ steps, and one typically needs to make many passes in order to complete a gradient descent. However, once the training is accomplished, the algorithm typically works rapidly, so training times can be amortized over the entire period of the use of the algorithm–training may be well worth it if you only need to retrain infrequently. Thus researchers that have been involved in using neural networks for clustering or classification have focused on reducing the number of features f that are used in grouping.

One of the best known ways of doing this is Latent Semantic Indexing (LSI) [29, 7], which applies the singular-value transformation (SVT) to the term-document matrix that characterizes a particular set of documents. (The SVT and its computation are discussed in [92] and [42]). The SVT returns (as one of its results) a diagonal matrix that has decreasing positive numbers on its diagonal, which are decreasing in size as one moves down the diagonal from upper left to lower right. One can approximate the original data by keeping only the larger of these singular values. This amounts to projecting the original vector space, in which the document vectors appear, onto a hyper-plane within that space (a lower dimension space).

The hyper-plane is selected by minimizing the sum-squared distance from the original points to their projections; thus maximal variability within the data is preserved. The dimensions of the new space are linear combinations of the original dimensions. These new dimensions are often thought as “conceptual” dimensions, because they tend to combine terms that tend to appear together. LSI’s developers claim that these new dimensions more accurately reflect the underlying semantic structure of the document while ignoring variability between documents in the use of particular terms. Documents that are close to one another in the new space can be clustered together to form a category.

Principal component analysis (sometimes called factor analysis) and the Karhunen-Loéve transformation are mathematically equivalent to the SVT. A problem with LSI and the SVT is that the document-term matrix tends to be very large, because there are typically many terms (there can be as many as appear in the dictionary!) and the cost of computing it is directly proportional to the size of the matrix. This may make it difficult to compute in real-time, if it is needed to produce the output of a search engine. However, once the LSI has been performed, it can be used to re-code all the documents at lower dimensionality, which, for instance, is useful for input to a neural network model for further clustering and classification, since many neural networks do not scale well to high dimensions. Thus it is necessary to reduce the dimensionality of the representation of the document vector before inputting it to a neural network model.

There are also other techniques used by researchers to reduce dimensionality. Often, they focus on finding the most discriminating terms in the documents.

For instance, Weigend, Wiener, and Pederson [122] use the chi-square statistic to find such documents. This statistic computes the (normalized) sum-squared difference between the expected occurrence rate of a term in each of a set of documents and the actual occurrence rate. Both low and high occurrence rates can be discriminating. Weigend et al. use this statistic to determine which terms are most discriminating for subsets of documents assigned to a particular topic. It is also common to eliminate very high-frequency words, such as “the,” because these are not likely to be useful in discriminating between documents. It is also common to eliminate very-low frequency words, because it not possible to gather good statistics on their appearance in documents.

In their work in text clustering, Sahami, Yusufali, and Baldonado [104] eliminate very high and very low-frequency words, and also eliminate a set of Web-specific words, such as “html” and “web,” before they do their clustering. Sahami, Hearst, and Saund [103] eliminate terms that occur in very few or very many of the documents of their corpus by using a Zipf’s law curve to eliminate low-frequency terms by fitting such a curve to the actual term frequency of a document and eliminating terms that occur after the curve falls off sharply. This has the advantage, they point out, of producing higher cutoffs for longer documents. Another approach taken to reducing the number of terms applies ideas used in the construction of decision trees in machine learning. Here, the terms that are most useful in separating out a particular topic are those terms that are best at discriminating that topic from the rest, those that are, in the lingo of decision trees, the most information content. Pazzani, Muramatsu, and Billsus [87] take this approach.

The HAL (Hyperspace Analogue to Language) model, developed by Lund and Burgess [17], computes, unlike the LSI model, a co-occurrence matrix between terms, that is, a term v. term matrix (LSI computes a term v. document matrix). The HAL model computes this matrix by using a sliding window that moves across a text. A typical window size is nine. The word that is in the center of the window is the term that has its components incremented. So, for instance, if the text contains the sentence "the quick brown fox jumped over the lazy dog," and the sliding window was centered on "jumped," the component for (jumped, over) would be incremented by 4, the one for (jumped, the) by 3, the one for (jumped, lazy) by 2, and the one for (jumped, dog) by 1. This is done similarly in the other direction: (jumped, fox) is incremented by 4, (jumped, brown) by 3, (jumped, quick) by 2, and (jumped, the) by 1.

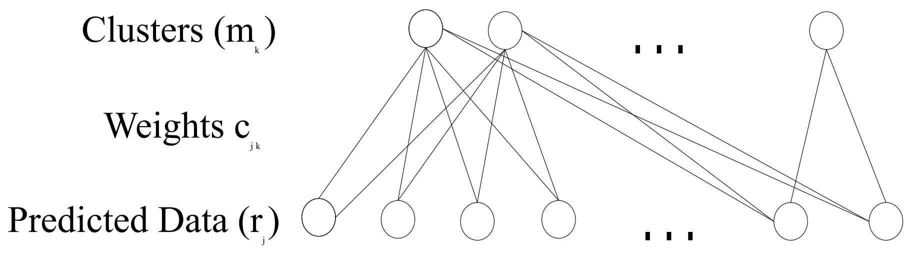
HAL's architects reduce the dimensionality of the distances between terms implied by their matrix (that is, the matrix values are interpreted as distances), using multidimensional scaling. One application of doing so can be for clustering. This clustering (onto a plane) has demonstrated that words that people judge as being similar are in fact closely grouped on this plane. Such clusterings could also be used to separate out search engine results, although HAL's architects have not done so.

2.9.3 The Multiple Cause Mixture Model for Document Clustering

One neural network model for clustering is Saund's Multiple Cause Mixture Model [103, 109]. This model is akin to a competitive learning model (e.g. [100]). Saund takes a causal view of document semantics, in which the underlying topic of each document is seen as causing the terms that appear in each document. Documents may have multiple topics and therefore have various mixes of terms resulting in the terms observed, hence the name "Multiple Cause Mixture." Saund claims that his model is different because of its many-to-many assignment of terms to topics, but this is not unique to his model, but exists in various versions of earlier approaches to document clustering, such those reviewed by Salton and Wong. Assignment of documents to multiple clusters or topics is typically a parameter of hierarchical grouping and iterative partitioning algorithms. However, Saund's model, although it has a similar architecture to many competitive learning models, is different from those models, in that in those models typically only one of the output units responds to a given stimulus, and they typically inhibit one another.

Also, the direction of the flow of activation in Saund's model is reversed; his formalism involves a vector m of "beliefs or activities" connected to a vector r of data value predictions by a matrix of weights c, as shown in Figure 2.2. All of the values in these variables fall between zero and one. The r vector is computed from the m vector and the c matrix by using a "soft disjunction," which is a fuzzy version of the logical OR function, also called the Noisy Or [88], designed to handle values between zero and one. This function was selected from among many possible candidates because it captures the idea that the presence of more than one topic (which here is a "cause") should only increase the propensity of a word related to each of those topics to appear. The causes are in the vector m, which represent the clusters; the results or predictions are in r, which is a term vector representing a document. The vector r represents the probabilities that individual words appear in a document, given particular cluster membership by having a single node active in m. The connections between a particular node in m and the vector r represent the cluster centroid; an "average" document in that cluster.

Figure 2.2: The Architecture of the Multiple Cause Mixture Model



The accuracy of the prediction vector r can be computed with a log likelihood function G involving r and the actual binary document vector d. The model is trained by doing gradient ascent to maximize G, which is the corpus-wide sum of the g value for each document. One node is added at the level of the m vector at time, starting with a random cluster centroid. After the first node is trained to a local maximum, it is split into two, and gradient ascent is continued until another local maximum is achieved. The algorithm stops when there ceases to be additional goodness-of-fit associated with additional cluster nodes. The gradient ascent is similar to the gradient descent used in back-propagation [101]; otherwise, the algorithm differs from back-propagation in a number of ways.

Since this is a neural network algorithm that uses gradiant ascent, it runs slowly, and is very sensitive to dimensionality. The authors used an approach drawing on Zipf's Law to eliminate some terms from the term vector representing a document and therefore reduce the dimensionality.

This algorithm is essentially unsupervised in nature (that is, it is a clustering algorithm), but it can be modified to be a supervised algorithm, which turns it into a text-classification algorithm rather than a text-clustering one. To do this, one performs a clustering, and then one associates each cluster manually with a class. Their experiments found that the algorithm performed respectively on classification tasks, but not as well as Naive Bayes. On a clustering task, it is more difficult to gauge performance; however, the authors do attempt to justify the clusters that the system formed on the basis of a particular test set.

2.9.4 Concept Indexing

Karypis and Han [56] propose a novel technique, that they call concept indexing, for dimensionality reduction in document retrieval. I mention it in this context (of clustering) because they use clustering to create new dimensions onto which they project the original documents. They use a version of a partitional clustering algorithm [1, 27, 65, 20] to create the initial k clusters that they cluster their document set into. The authors point out that partitional clustering algorithms have become popular in recent years because they have near-linear run-times, which are important in order to deal with very large document collections. In a direct partitional clustering algorithm, k documents are randomly selected as the seeds of clusters.

The remainder of the documents are then assigned to the cluster of the seed to which they are the most similar. The cluster centroid of each cluster is then computed. Then iterations are made through the documents, moving documents when they are closer to the cluster seed. One stops the clustering either after a fixed (small) number of iterations or when no document moves. So if the number of iterations is n and the number of documents in the collection is m, the time complexity of this algorithm is $O(nkm)$, where the individual operations performed are document-vector dot products that are limited in cost by the length of the longest document.

One can measure the quality of a clustering (relative to another one involving the same documents and the same number of clusters) by using the following metric M: the sum of the similarities of all of the documents to their respective cluster centroids. So the random seeding technique can be improved by running it several times with different sets of seeds and picking the clustering with the highest value of M. Of course, this "improvement" is purely technical, and may reflect nothing in terms of the subjective semantic quality of the clustering.

They use a variant of a random-seeded iterative partitioning algorithm to produce their clusters. After they produce their clusters, they use the cluster centroids as the axes of a new k-dimensional space by projecting the term vectors of the documents onto this space. They find that this dimensionality reduction has similar performance to Latent Semantic Indexing [29] at a much reduced computational cost, because there is no need to compute the Singular Value Decomposition. They also find that a concept-indexing-based dimensionality reduction improves the performance of standard classification algorithms such as C4.5 [93] and k nearest-neighbor (kNN) [74].

2.10 Document Classification

There have been a wide variety of techniques developed for the classification of textual documents. Most of these techniques do not rely on so-called "deep" knowledge about the domains of the documents in question, which would require true document and language "understanding" on the part of the machine, but rather rely on the statistical properties of such documents, mainly the distribution of the words in them. Nevertheless, researchers have been able to get good results in terms of percentages of documents correctly classified.

I engage in various experiments in the classification of Web pages in Chapters 3 and 4. See those chapters for more details.

Yang [128] reviews a variety of approaches to text classification, using the widely-used Reuters corpus of documents, which consists of financial news items from the Reuters news service. The Reuters corpus exists in several variants. She compared the CONSTRUE algorithm (an expert system developed by the Carnegie Group), a decision tree algorithm, a Naive Bayes algorithm, Inductive Rule Learning in Disjunctive Normal Form (DNF) (represented by two variants, the RIPPER and the SWAP-1 algorithm), two neural network algorithms (referred to as NNets.PARC and CLASSI), the Rocchio algorithm, the Linear Least Squares Fit algorithm, the Sleeping Experts algorithm, the k nearest neighbor algorithm, and the WORD algorithm, which does simple matching of the document in question with the category names, as a baseline benchmark of all the other algorithms.

I will briefly discuss some of the better-known of the algorithms Yang compared; for more details on these and the others, see Yang's paper and the references therein.

Decision tree algorithms, such as C4.5, build decision trees to classify examples based on an information gain metric from information theory. In the text classification context, where one is deciding if a document is in a category or is not, this means that the keyword that is best at distinguishing documents that are in a category from those that are not is used in the decision at the root of the tree.

The Naive Bayes algorithm, which is probably the most widely used text classification because of its combination of simplicity and effectiveness, uses the conditional probabilities of categories given the presence of a particular word, and then estimates the probability of category membership as the product of these probabilities, assuming that these events are independent (even though they clearly are not). Despite this simplifying assumption, one finds empirically that this method gives adequate performance, although, as we will see, Yang finds that it performs worse than several other techniques.

The neural network algorithms use either a two or three-layer feed forward network and gradient descent through parameter space [101]. Typically, one output unit is used for each category. Because these algorithms tend to run slowly, typically some method is used to reduce the dimension of the examples before the algorithm is run, such as Latent Semantic Indexing [29].

The k-nearest-neighbor algorithm, kNN, assigns a new example to the category of which the largest number of its k nearest neighbors are a member. Neighbors are typically defined in a Euclidian space with a Euclidian distance metric. Sometimes, a weighted distance metric is used [79].

Yang found that performance varied depending on the version of the Reuters database she used, and she did not test all algorithms with all variants. She measured performance based on the break-even point of a classifier, which reflects the classifier's performance when it has been tuned so its precision and recall are equal. Excluding the WORD algorithm, which gave poor performance on this metric (0.15 in the case of Reuters version 2 and 0.31 in the case of Reuters version 3), she found that performance varied from a low of 0.65 for Naive Bayes on Reuters version 2 to a high of .75 for EXPERTS, and a low of 0.71 for Naive Bayes on Reuters version 3 and a high of 0.85 for kNN. Thus Yang does not validate the commonly-held assumption that Naive Bayes is a well-performing algorithm for this problem. Statistical issues make the comparison of such algorithms a complex problem, though. Yang and Liu [127] updated Yang's work with another comparison of a smaller set of classifiers, and Naive Bayes did not do well in that analysis either. In that work, they found that Support Vector Machines and Linear Least Squares Fit, along with kNN, performed better than the neural network algorithms considered or Naive Bayes.

There are other text classifiers not considered by Yang. For instance, Lewis et al. [70] consider the performance of two classifiers, the Widrow-Hoff (WH) classifier and the Exponentiated-Gradiant (EG) classifier, and compare them to the well-known Rocchio algorithm. WH and EG are akin to neural networks, in that they use gradient descent in a parameter space of weights.

Chapter 3

Using Links to Improve Classification

Until about 1995, classification of textual documents into categories did not usually occur in a hyper-linked environment.[1] (For a comparison of some text classification techniques, see, for instance, Moulinier [83] or Dumais et al. [32]; also see my review in section 2.10). Thus, the text in each document was the only information that could be used to classify that document into two or more predetermined classes. In the Web environment, there are numerous hyper-links between documents. All things being equal, it seems reasonable that documents that are thematically related are more likely to be linked together than ones that are not on the same subject. Or, conversely (and probably more accurately) a document is likely to have links to similar documents on it.

We can make use of this fact to improve classification of documents. In fact, it is possible, as we will see, to improve classification performance based on link structure

[1]The term "hypertext" was coined by Ted Nelson in 1965, and other systems were hypertext-based before the Web; notably, Apple's Hypercard. However, until the advent of the Web, the use of hypertext systems was not sufficiently widespread to get much attention from the computer science research community. For Nelson's recent reflections, see [84].

alone, as compared to random classification.

3.1 Classifying Wisconsin Policy Documents Using Link Structure

3.1.1 Hypothesis

My hypothesis was that link structure among a set of classified examples could provide useful information about how these examples should be classified. The question, however, is how useful this link structure alone is. To measure this, in one context at least, is the purpose of this experiment.

3.1.2 Methods

I manually collected 500 documents (Web pages) on various aspects of public policy relevant to Wisconsin. I used publicly-available search engines, such as Altavista, Google, and Alltheweb to do this. One hundred documents were collected on each of the following topics (all with respect to Wisconsin): economy, education, the environment, government and politics, and health. In addition, 100 documents were collected on random topics, using Yahoo!'s random page retrieval facility, which selects a page from Yahoo!'s hierarchy (pseudo)-randomly. All of the documents that these 600 documents directly pointed to were also collected from the Web, which amounted to an additional 11,892 documents. This means that each of these 600 documents had an average of almost 20 links, but of course the number of links were not evenly distributed, with some pages having no or very few links, and some having many. (I also used the 500 classified documents in an experiment reported in Section 5.)

Since there are six categories, if we abandon all information about how they should be classified (which, of course, we know, since they were manually classified during my search process), and simply guess at their correct classifications, we will get the correct classification for one-sixth of the pages. If the sizes of the categories were different, one would get better performance by always guessing the most frequently occurring category, but here we have 6 categories with 100 pages each, so random guessing would work just as well.

I computed all of the 1-hop and 2-hop paths between these pages, ignoring any self-links (links from a page back to itself; all paths are directed). A 1-hop path is just a link; a 2-hop path is defined as a two-link path that gets you from the source to destination pages. Since there are 600 pages, there are $600 * 599 = 359,400$ possible directed combinations of these pages. However, there were only 46 1-hop paths and 215 2-hop paths found among these pages. So, this part of the Web is very sparse. As a percentage of the number of pages, the one-hop paths represent (46/600) or about 0.076 in-links per page, and the two-hop in-links represent (215/600) or about 0.358 in-links per page.

For each of the 600 pages in the original set, we can consider what the link information tells us about what classification they should have. Of course, in this case, we already know this information, but the algorithm that I describe below can also be used on pages that are not in the original set of pages, such as pages close to them in the graph that is the Web.

The following simple algorithm can be used. For each page and for each category,

Figure 3.1: Voting Algorithm for Classification of Pages by Direct and Indirect Incoming and Outgoing Links

```
1. For each page $p$:
   (a) For each category $c$:
       i. compute number of pages $n_{in,1,p,c}$ in $c$ which directly point to $p$.
       ii. compute number of pages $n_{in,2,p,c}$ in $c$ which indirectly point to $p$ (via another page).
       iii. compute number of pages $n_{out,1,p,c}$ in $c$ to which p directly points.
       iv. compute number of pages $n_{out,2,p,c}$ in $c$ to which p indirectly points (via another page).
       v. Let $n_{p,c,total} = n_{in,1,p,c} + n_{in,2,p,c} + n_{out,1,p,c} + n_{out,2,p,c}$.
   (b) Assign $p$ to the category $c$ with the maximal value of $n_{p,c,total}$.
```

sum: the number of pages in that category with a 1-hop path to that page, the number of pages in that category with a 2-hop path to that page, the number of pages in that category with a 1-hop path from that page, and the number of pages in that category with a 2-hop page to that page. After computing these statistics for all pages and categories, classify each page in the category that has the largest sum.[2] This algorithm is laid out in detail in Figure 3.1. The algorithm is a variant of a "leave-one-out" training-test set regime, in which a training set of size k (here, 600) is divided into a training set of size *k-1* and a test set of size one; all the other instances are used to test each instance [79][82].

[2]It would probably be an improvement to give a higher weight to the direct as opposed to the indirect links in this algorithm, because direct links are both less frequent and probably on average more meaningful. However, I have not done so here; in order to do so, an appropriate weighting factor would have to be selected; one possibility would be a ratio computed by finding the number of direct versus indirect (two-hop) links in a random sample of Web pages.

3.1.3 Results and Discussion

Table 3.1 is a "confusion matrix" that shows the classification of pages in the original 5 categories into the same 5 categories based on this algorithm. The original category runs down the vertical axis, and the new category runs across the horizontal axis. The final column shows the number of pages that are classified at all using this method (which for all categories is a minority). If a page has no paths in or out of it from other pages, then it cannot be classified using this method, and due to the sparseness of the graph, many pages fall into this category. Only the pages that can be classified in such a manner are showed in this table; however, all pages are accounted for in the overall classification performance statistics shown in Table 3.2, by assigning pages that have no information uniformly across categories (since, here, the categories are of uniform size). However, Table 3.1 is useful in order to see the proportion of pages for which this algorithm is useful.

Note that the final category (6) consists completely of random pages. Each of these pages points to no other page of the 600, nor is it pointed to by any of them. So this classification method does not work at all on these pages; none of them is classified into any category, and they are omitted from Table 3.1. However, we can see from this table that except for category 1, more pages in each category are classified back into that same category using this method, since for categories 2-5, the maximal values are on the diagonal. Thus, this demonstrates that at least for this data set, the link information in combination with the category information for nearby pages can provide additional information as to a page's classification. Note that since there are equal numbers of pages in each category in this example, we do not need to worry about correcting for the (more general) situation where some categories have more pages in them than others, and therefore, all things being equal, provide more out-links.

Table 3.1: Confusion Matrix for Classification of Wisconsin Public Policy Web Pages

	Reclassified Category					
Original Category	**1**	**2**	**3**	**4**	**5**	**Totals**
1	5	1	1	11	0	18
2	1	24	2	6	1	34
3	3	3	23	5	0	34
4	3	0	3	41	1	48
5	2	2	0	9	21	34

This algorithm will only work well if there is some semantic separation between the categories that is reflected in the link structure. We can see from the table above that category 4—which is "politics and government"—is intertwined semantically, and in terms of the link structure, with the other categories. Pages in category 4 are more entwined with pages in category 1 ("economy") than category 1 pages are entwined with other pages in category 1, which is not tremendously surprising given the close relationship between these two topics. The degree to which this happens will depend on the underlying semantics of the human classification that is being used.

Table 3.2 shows the percentages of pages classified in each category, assuming that one-sixth of each of the pages not classified by the method above are assigned to each category. For categories 1 through 5, the value on the diagonal is greater than it would be (16.7 percent) with no additional information, since the categories are of equal size, and if we guess randomly among the six categories.

Table 3.2: Confusion Matrix, Classification of Wisconsin Policy Documents, Percentages

	Reclassified Category					
Original Category	**1**	**2**	**3**	**4**	**5**	**6**
1	18.7	14.7	14.7	24.7	13.7	13.7
2	12.0	35.0	13.0	17.0	12.0	11.0
3	14.0	14.0	34.0	16.0	11.0	8.7
4	11.7	8.7	11.7	49.7	9.7	11.0
5	13.0	13.0	11.0	20.0	32.0	16.7
6	16.7	16.7	16.7	16.7	16.7	16.7

3.1.4 Related Work

Others (notably Tom Mitchell and his colleagues, e.g. [11][85][86]) have made use of link information in Web document classification. However, their main concerns have been with using the information on the anchor tag (the text on the link) pointing into a Web page, and on using techniques to bootstrap from a small set of labeled examples and a larger set of unlabeled examples to successfully classify a larger set of pages. Here, I am doing something simpler. I have simply demonstrated that the classification of the immediate and two-hop neighbors of a page in either direction along the directed graph of the Web can provide us with some useful information about the classification of that page. This leads up to the result of Section 3.3 that combining such link information with the textual information on page as analysed by the Naive Bayes algorithm, we can attain better classification performance than with either one separately.

3.2 Link Structure and Classification in a Sample of Pre-Classified Documents

One of the problems with the exercise described above in Section 3.1 is that I did the classification of the documents, and it is likely that I am biased. So, as an antidote to this, I selected part of the tree of pre-classified documents found in the Open Directory (Dmoz, or "Mozilla Directory") project site at www.dmoz.org. This is a site in which a large number of volunteer "editors" classify pages into a hierarchical taxonomy, like a library's card catalog, and the results of this classification are made freely available for browsing and for download (the latter in a huge file).

I selected all the Web pages that the Open Directory editors had put in the category "/Science/Biology." At the time that I downloaded these pages, there were 12,828 pages in this category. Of these, nine of the pages were classified at the root, and the remaining 12,819 in one of the 32 sub-categories below the root. These pages were *not* distributed uniformly in these categories; they ranged from 8,140 pages classified into "/Science/Biology/Taxonomy" to 2 pages classified into "/Science/Biology/Associations".

There were a total of 64,544 links between pages in "/Science/Biology;" however, 61,875 of these links were between pages in "/Science/Biology/Taxonomy;" this left 2,669 links in which at least one of the categories was not "/Science/Biology/Taxonomy." Considering all links, this is $65,544/12,828$ or about 5.1 in-links per page; considering just the links not in the taxonomy clique, this is $2669/12,819$ or 0.2 in-links per page. The Wisconsin Policy collection had about 0.08 in-links per page, which is on approximately the same order of magnitude as the latter (0.2) figure. The Web has many sets of pages that have been manually constructed with large cliques; in fact, both Yahoo! and Dmoz contain such cliques. Since different samples of pages may vary significantly on this dimension, this may significantly affect the average number of in-links found in any particular sample.

I constructed a 32-by-32 array representing the links, where the vertical axis was the category of the source page and the horizontal axis was the category of the destination page. The value in each array cell is the number of links falling into that cell. This array is shown in Figure 3.2.

If a page in a given category was more likely to point to other pages in that category than to pages in other categories, than one would expect the diagonal values in this array to be maximal; that is, to be greater than all of the other values in each diagonal value's column. This is the fact the case for 19 of the 32 categories in the figure. Of course, if the maximal values were randomly distributed in each column, very few of them would be on the diagonal. Thus, the fact that most of them are on the diagonal is good evidence that links coming from pages that have already been manually classified can provide good information about the category membership of the pages to which they point.

Figure 3.2: Counts of Linkages Between Categories in the Dmoz "/Science/Biology" Subtree

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	63	7	0	2	10	0	13	0	4	6	0	2	1	1	22	0	0	2	4	0	14	0	1	0	1	0	0	0	1	0	0
3	0	11	12	0	0	0	0	0	0	0	2	0	0	0	0	15	0	0	1	2	0	8	0	0	0	2	0	0	0	0	0	0
4	0	2	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	21	4	1	1	0	0	3	1	1	4	0	5	0	0	0	2	0	2	0	0	0	1	0	0	0	0	0	0
6	0	4	0	0	1	76	0	1	0	1	1	5	0	1	0	4	0	0	0	1	0	4	0	0	0	0	0	0	82	0	0	2
7	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	1	1	0	3	0	0	11	0	6	4	0	0	2	0	7	0	0	0	5	0	4	0	0	0	0	0	0	0	0	0	3
9	0	0	0	0	0	0	0	0	74	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	0	0
10	0	2	0	0	0	0	0	2	0	7	4	0	1	0	0	19	1	0	0	2	0	0	0	2	0	0	0	0	0	0	0	3
11	0	3	1	1	3	3	0	1	0	2	2	4	10	1	0	4	0	0	1	4	0	3	0	1	0	3	0	0	1	0	0	7
12	0	0	0	0	0	9	0	0	0	1	0	99	1	0	0	1	0	0	0	1	0	7	0	0	0	1	0	1	2	0	0	13
13	0	0	0	0	0	3	0	0	0	1	2	3	9	0	1	5	0	0	0	0	0	5	0	2	0	2	0	0	1	0	0	4
14	0	2	0	0	1	0	0	3	0	0	2	0	1	8	0	0	0	0	3	2	0	2	0	1	0	0	0	0	0	0	0	0
15	0	16	12	0	13	1	1	5	0	5	1	3	0	2	15	24	0	0	5	1	0	8	0	1	1	2	1	2	4	4	0	1
16	0	6	3	0	18	5	0	3	0	6	2	0	1	0	1	63	0	0	0	4	0	6	0	3	0	1	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	2	2	0	0	0	1	0	0	1	2	0	3	0	0	17	3	0	6	1	0	0	0	3	0	0	0	1	0
20	0	11	0	0	0	1	0	7	0	0	2	1	0	0	0	1	0	0	0	5	0	3	0	0	0	0	0	0	0	0	0	2
21	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
22	0	2	2	0	0	2	0	2	0	2	0	1	2	2	1	6	0	0	3	2	0	61	2	0	0	2	0	0	5	0	0	6
23	0	1	0	0	0	2	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
24	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	3	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
26	0	2	1	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0
29	0	1	1	0	0	57	0	1	0	0	0	29	1	0	2	0	0	0	0	0	0	3	10	0	0	0	0	0	61875	0	0	43
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
31	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
32	1	1	1	0	0	1	0	0	0	0	3	14	1	4	1	2	0	1	0	4	0	5	0	0	1	0	1	1	56	0	0	1089

3.3 Extending Naive Bayes to Use Hyper-link Information

3.3.1 Hypothesis

The hypothesis explored in this section is that combining textual and link information can do a better job of classifying Web pages than either one alone.

3.3.2 Methods

I explore six different algorithms and compare their performance; they all incorporate both textual and link information, except for text-based Naive Bayes, which I include as a baseline by which all the others are measured. They are described as follows.

One commonly-used and effective way to classify text uses the Naive Bayes method, given by the following formula:[3]

$$nbscore_T(d_i, v_j) = P(v_j) \prod_{k \in W} P(w_k \mid v_j)$$

where v_j is one of the set of possible classifications $(v_1, \ldots, v_n)$ and the w_k are the words drawn from the word positions W found in document d_i of the set of documents being classified. $P(v_j)$ is the overall probability of category v_j. $P(w_k \mid v_j)$ is the probability that w_k is in the document, given that the document is in category v_j. These probabilities are estimated from a training set of documents, and then documents in a test set are classified. The subscript T stands for "text." The document d_i is classified in the category v_j that has the highest value for $nbscore_T(d_i, v_j)$. For the

[3]This formula was taken with slight notational, but not substantive, changes from Mitchell [79].

Figure 3.3: The Naive Bayes (*NB*) Algorithm, as Applied to Text Classification

1. Divide the sample into a training set and a test set.
2. Let W be the set of words w_k found in any document in the training set. Let n be the total number of word positions in all the documents of the training set concatenated together. Let n_{w_k} be the total number of times word w_k appears in the training set in all documents classified in category v_j. Then let
$$P(w_k \mid v_j) = \frac{n_{w_k} + 1}{n + |W|}$$
3. For each document d_i in the test set and all w_w in W in that document:
   (a) For each category v_j in the set of possible classifications let
   $$nbscore_T(d_i, v_j) = P(v_j) \prod_{k \in W} P(w_k \mid v_j)$$
   (b) Assign d_i to the category v_j with the highest value of $nbscore_T(d_i, v_j)$.

purpose of this discussion, let us refer to this algorithm as *NB*; this is the ordinary Naive Bayes text classification algorithm widely used in the literature. A pseudocode version of this algorithm is given in Figure 3.3.

In an environment, such as the Web, where there are directed links between documents, it is reasonable to believe that if a document links to and is linked to from other pages known to be in a particular category, that it is more likely to be in this category, all other things being equal. It is a straightforward matter to extend Naive Bayes to make use of such link information. We simply add more conditional probabilities to the formula. Now we have

$$nbscore_{TL}(d_i, v_j) = P(v_j) \prod_{k \in W} P(w_k \mid v_j) \prod_{l \in C} P(a_{iv_l} \mid v_j) \prod_{l \in C} P(b_{iv_l} \mid v_j)$$

where the subscript TL signifies "text and links," a_{iv_l} is the number of in-links to document d_i from pages in category v_l, b_{iv_l} is the number of out-links from document d_i to pages in category v_l, and C is is the set of categories in which documents are being classified. $P(a_{iv_l} \mid v_j)$ is the probability that the document has a_{iv_l} in-links from category v_l, given that the document is in category v_j. $P(b_{iv_l} \mid v_j)$ is a similar term with respect to the number of out-links. Note that these additional conditional probabilities are estimated using the training set, as are the word occurrence conditional probabilities. Let us refer to this algorithm as *NB-Text-And-Links*. A pseudocode version of this algorithm is shown in Figure 3.4.

Another way to extend Naive Bayes is to use additional "bags of words" [22]. For instance, if we consider a set of pages in a particular category that have been divided into a training set and test set, we can consider the first bag for a given example (page) to be all the terms drawn from the text on that page, and the second bag to be all the words drawn from pages that are in the training set and point to or are pointed to by the page that is the example. (In my experiments, I use pages that are one or two hops away. My algorithm varies a bit from that used in [22], in that the authors in that paper use three bags, one for the text of the example, one for inlinks, and one for outlinks, and they only use direct links, while I use 1 or 2-hop connections.) Each of these bags can be considered separately, forming its own product with respect to each of the categories v_j above. The same term appears twice if it appears in both bags, and each instance is considered different for the purposes of the algorithm. The scores for the two bags can be multiplied together and the result used as the final product for classification of a test example. Let us refer to this algorithm as *NB-2bags*. It is described in pseudocode in Figure 3.5.

Figure 3.4: Algorithm *NB-Text-And-Links*: Integrates Textual and Link Information to Classify Pages

1. Divide the sample into a training set and a test set.
2. Estimate the probabilities $P(w_k \mid v_j)$, the chances of the presence of word w_k, given that the document in question is in category v_j, using word frequencies in the training set as in the Naive Bayes algorithm: see Figure 3.3.
3. Let $P(a_{iv_l} \mid v_j)$ is the probability that a document d_i has a_{iv_l} in-links from category v_l, given that the document is in category v_j, and $P(b_{iv_l} \mid v_j)$ is a similar term with respect to the number of out-links. $P(a_{iv_l} \mid v_j)$ and $P(b_{iv_l} \mid v_j)$ are estimated using by counting links within the training set.
4. For each document d_i in the set of documents in the test set:
   (a) For each category v_j in the set of categories *C:*
      i. For each category v_l in the set of categories C let

      $$nbscore_{TL}(d_i, v_j) = P(v_j) \prod_{k \in W} P(w_k \mid v_j) \prod_{l \in C} P(a_{iv_l} \mid v_j) \prod_{l \in C} P(b_{iv_l} \mid v_j)$$

      where the subscript TL indicates "text and links," where w_k is a word in the set of words W found in the document.

   (b) Assign d_i to the category v_j with the highest value of $nbscore_{TL}(d_i, v_j)$.

Figure 3.5: Two-bag Version of Naive Bayes (*NB-2bags*)

1. Divide the sample into a training set and a test set.

2. Let W be the set of words w_k found in any document in the training set. Let n be the total number of word positions in all the documents of the training set concatenated together. Let n_{w_k,v_j} be the total number of times word w_k appears in the training set in all documents classified in category v_j. Then let, for all v_j,
$$P_1(w_k \mid v_j) = \frac{n_{w_k v_j} + 1}{n + |W|}$$

3. For each document d_i in the training set, construct a new document $d_{i,neighbors}$ which consists of all the neighbors of d_i one-hop or two-hops away in either direction, concantenated together. Redefine n as the total number of word positions in the $d_{i,neighbors}$ concatenated together (this will reuse documents). Redefine $n_{w_k v_j}$ as the total number of times word w_k appears in $d_{i,neighbors}$ for all documents d_i classified in category v_j. Then let, for all v_j,
$$P_2(w_k \mid v_j) = \frac{n_{w_k v_j} + 1}{n + |W|}$$

4. For each document d_i in the test set and all w_k in W in that document and all w_l in $d_{i,neighbors}$ corresponding to d_i (as in step 3 above, and using only neighbors in the training set):

   (a) For each category v_j in the set of possible classifications let
   $$nbscore_T(d_i, v_j) = P(v_j) \prod_{k \in W} P_1(w_k \mid v_j) \prod_{l \in W} P_2(w_l \mid v_j)$$

   (b) Assign d_i to the category v_j with the highest value of $nbscore_T(d_i, v_j)$.

Figure 3.6: Algorithm *Simple-Link-Voting*: Classifies Pages on Basis of Link Information Only

```
1. Divide the sample into a training set and a test set.
2. For each document $d_i$ in the test set:
    (a) For each category $v_j$, let $n_{v_j}$ be equal to the total number of direct (one-hop) and indirect (two-hop) in-links and out-links from pages in the training set that have been classified in category $v_j$
    (b) Assign $d_i$ to the category $v_j$ with the highest value of $n_{v_j}$.
```

Let us also consider a simple algorithm based only on links, as a baseline, and to see whether link information alone can provide significant information for classification. Simply, for each page, find the category for which there are the most number of pages in the training set in that category pointing to that page or which that page points to (through 1 or 2 hops), and select this category as the category in which to classify the page. Let us refer to this algorithm as *Simple-Link-Voting*, since we can think of the links as "votes." This algorithm is shown in pseudocode in Figure 3.6.

Let us also consider a couple of variants on the above. Algorithm *Voting-Trumps-NB* allows the selection of *Simple-Link-Voting* for the category membership of a page to trump (that is, have its value supersede that of) *NB*, unless the former provides no category with any "votes" (which often happens due to the sparse nature of the connectivity matrix). The pseudocode for Voting-Trumps-NB is shown in Figure3.7.

Algorithm *Voting-NB-Combined* lists the top three category choices of *Simple-Link-Voting* and *NB* and sums the ranks for each category, picking the category in the list with the lowest summed rank as a way of combining the two rank order lists. If a category appears on only one of the lists, its score is doubled for comparability.

Figure 3.7: Algorithm *Voting-Trumps-NB*

1. Divide the sample into a training set and a test set.
2. For each document d_i in the test set:
    (a) Assign d_i to the category given by *Simple-Link-Voting* (see Figure 3.6), unless *Simple-Link-Voting* offers no category assignment, in which case assign d_i to the category given by Naive Bayes (see Figure 3.3).

It is shown as pseudocode in Figure 3.8.

Algorithm *NB-with-Neighbors* applies Naive Bayes to the page to the page to be classified and to all the pages to which it points (retrieving these neighbors from the Web if necessary; that is, if they are not already in the test or training sets). It then counts "votes" from all these pages and the page itself, assigning the page to the category that receives the most votes. This algorithm is shown in pseudocode in Figure 3.9.

For the purposes of comparing the above algorithms, I selected pages from the "/Science/Physics" and "/Science/Chemistry" categories of Dmoz. In the former set, there were 1,782 pages, and in the latter set, 1,379.[4] The chemistry pages were divided by Dmoz into 24 categories, and the physics pages into 27 categories. Classification into these categories was the task for the above algorithms. Random classification would lead to low classification performance. In the case of physics, if you assume equal classification probability for each of the 27 categories, you get $1/27 = 3.7$ percent classified correctly; if you always guess the most frequently-occurring category, you

[4]Actually, these numbers were 1,899 and 1,472 originally, in terms of the list of URLs cataloged in Dmoz but a fraction—between 6 and 7 percent of the pages in each case—could not be retrieved from the web, due to the fact that pages tend to disappear from the web after Dmoz catalogs them, or are unavailable due to server or network down time.

Figure 3.8: Algorithm *Voting-NB-Combined*

1. Divide the sample into a training set and a test set.
2. For each document d_i in the test set:
   (a) Find the categories v_{11}, v_{12}, and v_{13} with the three highest values of n_{v_j} for d_i (in order) as given by *Simple-Link Voting* (see Figure 3.6).
   (b) Find the categories v_{21}, v_{22}, and v_{23} with the three highest values of $nbscore_T(d_i, v_j)$ (in order) as given by *NB* (see Figure 3.3). Note that some of these may be equal to the categories in (a).
   (c) Let the rank of v_{ik}in (a) and (b) above be k.
   (d) For each unique category v in the (up to) six listed above in (a) and (b):
      i. Sum its ranks k to form $S_{v.}$ If v appears in only one list, add that rank into the sum twice, for comparability.
   (e) Assign $d_i$to the category v with the lowest summed rank S_v.

Figure 3.9: Algorithm *NB-with-Neighbors*

1. Divide the sample into a training set and a test set.
2. Define $P(w_k \mid v_j)$ as in Naive Bayes using the training set (see Figure 3.3).
3. For each document d_i in the test set, find all of the pages d_j to which it points.
   (a) Form the union of d_i and all the d_j. Call this D. Initialize $count(v)$ to zero for all categories v. For each d_k in D:
      i. Find the category v that it would be assigned by the Naive Bayes algorithm (see Figure 3.3). Increment $count(v)$ by one.
   (b) Assign d_i to the category v with the maximal value of $count(v)$.

get 14.1 percent. For chemistry, these numbers are 1/24 = 4.2 percent and 30.3 percent. The numbers for guessing the most frequent category provide a baseline for the classification performance of all the other algorithms.

Each of the chemistry and physics sets were separated randomly into 10 equal-size "folds" for k-fold cross-validation [79], with k=10.[5] In this technique, which is commonly used, each fold becomes the test set in one iteration of each algorithm, and the union of the other 9 folds becomes the training set. The mean performance across the 10 test sets is used to access the performance of a particular algorithm, in order to get a better estimate of the true performance of the algorithm. In addition, a paired t-test can be used to statistically compare two algorithms, by pairing up the performance statistics of the two algorithms on each particular fold. Twenty data points are used for each of these paired t-tests, 10 pairs of two each.

3.3.3 Results

The mean performance of all the algorithms described above on these two data sets is given in Table 3.3. Also shown is the difference in this mean performance from that of Naive Bayes and the significance of this difference as measured by a two-sided t

[5]All statistical methods for accessing the differences in the performance of learning algorithms are prone to some Type I error. Type I error occurs, in this case, when such a method incorrectly reports a difference between two learning algorithms' performance as being statistically significant. Dietterich [30] compares approximate statistical techniques for use when comparing machine learning classification algorithms. He finds that 10-fold cross-validation, compared to some other statistical techniques, has "somewhat elevated" probability of Type I error (on real-world data, this amounts to a probability of between five and ten percent, depending on the data set in question); however, it is also the most powerful in its ability to detect differences between algorithms when they do exist. In cases when the algorithms can be run more than once, as is the case with the algorithms that I discuss in this section, he recommends the use of another statistical technique, which not as powerful as k-fold cross-validation, but has a lower probability of Type I error (around four or five percent in his tests). This latter technique is 5x2 cross-validation, in which cross-validation is run five times, each with two equal-sized, randomly-selected folds.

test and its corresponding p value, which is the probability that this difference could have occurred. Here, there are 9 degrees of freedom.

Table 3.3: Mean Performance Across Ten Folds for Chemistry and Physics Data Sets and Significance of Difference from Naive Bayes as Measured by a Paired Two-Sided t Test

Chemistry Data Set, 10-fold Cross-Validation				
Algorithm	Mean test set perf., folds	Diff. from *NB*	t value	p value
NB	0.450	0	N.A.	N.A.
NB-Text-and-Links	0.472	0.022	7.043	0.000
NB-2bags	0.443	-0.006	-0.624	0.548
Voting-NB-Combined	0.459	0.009	1.159	0.276
Voting-Trumps-NB	0.558	0.108	6.152	0.000
Simple-Link-Voting	0.409	-0.041	-1.628	0.138
NB-with-Neighbors	0.393	-0.056	-3.529	0.006

Physics Data Set, 10-fold Cross Validation				
Algorithm	Mean test set perf., folds	Diff. from *NB*	t value	p value
NB	0.409	0	N.A.	N.A.
NB-Text-and-Links	0.424	0.015	2.907	0.017
NB-2bags	0.404	-0.005	-0.488	0.637
Voting-NB-Combined	0.481	0.072	5.239	0.001
Voting-Trumps-NB	0.573	0.164	7.575	0.000
Simple-Link-Voting	0.515	0.106	4.233	0.002
NB-with-Neighbors	0.305	-0.104	-9.379	0.000

3.3.4 Discussion

The first thing to notice about Table 3.3 is that all of the test set classification percentages are considerably higher than would be expected by chance, and all are higher than what would be expected by simply guessing the most frequently-occurring category, which is a good thing (but not surprising, since we know that *NB* performs much better than chance in general). This is a relatively difficult task for the classifiers due to the large number of possible classes, so the test set performances are relatively low. *Simple-Link-Voting*, a very simple algorithm, gives valid classification information over and above that which one would obtain at random, and does quite a good job in the case of the physics pages. *Voting-Trumps-NB* performs best; 16 points better than NB for the physics pages and 11 points better for the chemistry pages. In addition, these differences are very highly significant, with p values approaching zero.

For the chemistry pages, *NB-Text-and-Links* and *Voting-Trumps-NB* are the only algorithms that performs significantly better than *NB*; *NB-with-Neighbors* performs significantly worse. The rest have no significant differences from NB, as measured by the p value. For the physics pages, three of the algorithms (*Voting-NB-Combined*, *Voting-Trumps-NB*, and *Simple-Link-Voting*) perform significantly better than *NB*; *NB-with-Neighbors* performs significantly worse. All of these differences are highly significant as measured by the p value.

Thus, of all of these variants, *Voting-Trumps-NB* and *NB-Text-and-Links* are the only ones that give a consistent significant improvement over *NB*, and *Voting-Trumps-NB* gives performance that is much better. This would seem to indicate that the best approach is that when you have significant in-link or out-link information that indicates a particular category, than use that information in preference to that provided by the (text-based) *NB*; if you are (as you often are, given sparse networks) lacking such information, use the *NB* information. Of course, the amount of such information that you have is roughly proportional to the size of your training set, assuming a fixed average number of in-links per page. So, very little additional information is obtained with small labeled sets, and improvement should be better with bigger labeled sets.

Chakrabarti, Dom, and Indyk [22] argue that *NB-2bags* performed poorly in their experiments because links from and to pages point to pages on a diverse set of topics, tending to dilute the efficacy of training text in discriminating a particular topic. Unlike them, I found no significant difference in performance between *NB* and *NB-2bag*s. However, I think that their argument accounts for the poor performance of *NB-with-Neighbors*, which was the only algorithm to do significantly worse than *NB* on both the chemistry and the physics sets. It is better to concentrate on textual information simply on the page itself, this would indicate. Manually-assigned labels on neighbors are useful; labels on neighbors assigned by *NB* are not, at least when they are used in the manner of *NB-with-Neighbors*. However, my experiments provide evidence that neighbor information, when used in the manner used by *Voting-Trumps-NB* or, less effectively, *NB-Text-and-Links*, is quite useful in providing improvement over *NB* alone.

This evidence is a contribution to the literature that can be made use of in the following manner. Given the large numbers of pages that have been preclassified manually via Dmoz or Yahoo!, this should have a good deal of practical utility, especially in the classification of those pages' unclassified neighbors. A very large number of pages are one or two links away from one or more pages in Dmoz or Yahoo!. This is because Dmoz and Yahoo! are large and tend to contain the most highly referenced pages on the Web—often "hubs" or "authorities" in the terminology of the

For each of these pages in Dmoz or Yahoo!, the fact that *Voting-Trumps-NB* is

Table 3.4: Information Used Versus Techniques Employed for the Algorithms Discussed in this Section; Original Algorithms Shown in Bold Italics; Algorithms from the Literature, in Ordinary Italics

	Naive-Bayes-Based	**Voting-Based**	**Both Naive-Bayes and Voting-Based**
Labels on Neighboring Pages		***Simple-Link-Voting***	
Text on Page	*Naive Bayes*		
Text on Page and Text on Neighboring Pages	*NB-2Bags*		***NB-with-Neighbors***
Text on Page and Labels on Neighboring Pages	***NB-Text-and-Links***		***Voting-NB-Combined; Voting-Trumps-NB***

the best performing algorithm of the set of algorithms I have considered indicates that one should classify the page into the most frequent class of the set of 1-hop and 2-hop neighbors in preference to the class given by *NB*. This seems to be an important contribution of this section: this result can be used to guide the activity of a spider that is classifying pages as well as the building the usual reverse-keyword index for a search engine. If user feedback is collected about a provisional automatic classification of a page into a category, this could allow categories to grow in a semi-automated fashion. If user participation could be solicited in this manner (that is, if users could be solicited to validate pages that had provisionally been placed into classes), allowing classes to grow, training sets could also grow, allowing even more pages to be classified more accurately.

Table 3.4 illustrates the information used by each of the learning algorithms discussed in this section versus the techniques that they employ. The two algorithms (*Naive Bayes* and *NB-2Bags*) that I took from the literature are shown in ordinary italics; the five variants that I devised are shown in bold italics. It is not surprising that the most successful technique, *Voting-Trumps-NB*, employs both the most information and both techniques. As this table illustrates, I have not fully explored the space of possible variations on these algorithms; however, most of the remaining empty cells would require that I devise a method to apply a voting-based technique to textual information. The upper-left cell in the table could be readily filled by using the link information in the manner used in *NB-Text-and-Links*, but without the textual information.

In related work, Chakrabarti et al. [22] use a relaxation labeling technique to incorporate link information into classification. On some of their test beds, they get significant improvement in classification performance over text classification that does not use link information. Sean Slattery (e.g. [115]) has done work on applying relational learners, which learn sets of rules, such as FOIL [94], to Web page classification, while incorporating link information using the Clever project's [58] work on hubs and authorities.

3.4 Review of the Contributions of this Chapter

This chapter has contributed to the literature on the relative importance of classification information gathered from neighbor pages as opposed to text on a page itself. In the "small world" context of the Web, in which large manual classifications exist precisely of the those pages that are most likely to be referenced by others, this neighbor information is likely to be crucial in improving classification performance of those many pages which are neighbors of the ever-growing set of pre-classified pages in human-maintained hierarchies such as Dmoz or Yahoo!.

In Sections 3.1 and 3.2, I demonstrated that in two different contexts, Web pages in a particular category are much more likely to point back to other pages in that category than to other categories, by using a simple, direct voting algorithm. These sections exist mainly as motivation to Section 3.3, to make the prima facie case that link information will be useful in supplementing textual information. This point, of course, has been made elsewhere (e.g. [23]), but I have not seen it done in such a direct manner. Sections 3.1 and 3.2 demostrate this in two more domains, and in a simple, direct, manner, using a simple voting algorithm and confusion matrices.

In fact, Section 3.3 makes two contributions to the literature. The first contribution is the demonstration that, on two datasets, neighbor classification information alone, when available, should be used in preference to texual information alone; that is, in the technical terms outlined in the section, that $Voting-Trumps-NB$ significantly outperforms NB. I also contribute data on the performance of 4 other variants of NB and one algorithm involving voting alone.

The second contribution is more practical; if the number of neighbors (that is, the size of the labelled set) was static and the graph of the Web was random and uniform, the result that $Voting-Trumps-NB$ significantly outperforms NB would have little import. But, in fact, the size of the labelled set is constantly growing, through the efforts of the builders of taxonomies such as Yahoo! and Dmoz. And, the Web is by no means anywhere close to a random, uniform graph. It has pages that have wildly disproportionate numbers of out-links or in-links ("hubs" or "authorities," respectively in the terminology of the Clever project [21, 58]). Because of factors like this, the Web is a "small world" network connected through such highly referenced or referencing pages—as I have discussed in Section 2.1. And these hub and authority pages are much more likely to be listed in the manual directories like Yahoo! and Dmoz. The fact that $Voting - Trumps - NB$ outperforms NB is therefore a significant result, because, for instance, if a page X is listed on a hub page that has been classified under "Biochemistry," or points to an authority page that has been classifed under "Biochemistry," this means that one is better off classifying page X under "Biochemistry" as well, as opposed to classifying it on the basis of the text on the page.

If user feedback can be gathered, then users interested in a particular category should first be queried about unclassified pages with the highest number of neighbors in that category, and then the process could be continued for pages with fewer neighbors. The problem, in the Web context, is not having enough information; it is in filtering it properly, and this gives users a way to prioritize their activity. It also could assist the users who manually construct the taxonomies, in that pages could be fed to them in an prioritized, automated fashion for acceptance or rejection.

Chapter 4

Merging Search Engines and Directories

There is no principled reason why search engines and directories need be two separate entities. I believe that the reason that they have evolved independently on the Web is mainly the result of the social history of the Web and of prior interfaces to textual information, in both electronic and paper form, rather than any explicit engineering or human interface need for them to have developed in this fashion. Search engines work like the full-text databases, such as Lexis/Nexis, that predated the Web; Web directories are based on library card catalogs, although they are actually not as sophisticated in some ways. Thus, there has been little imagination used in construction of new interfaces to massive quantities of text since the Web began.

Starting with a hierarchical classification of Web pages—I have used the pages of the Open Directory Project (at www.dmoz.org) because of its large size and non-proprietary nature—it is possible to build a search engine/directory that combines the best features of both, and in fact has emergent features which improve on either one on its own. The SONIA system [104] is another system which combines clustering and classification, but the architects of this system have not emphasized the close affinity of page ranking with collaborative filtering, and the ability to use such filtering to build communities within the information system.

The idea is that every page, as it is encountered by a spider, would both be classified in a category within a directory, and also its text would be indexed in a reverse-keyword index. A searcher could search using a keyword phrase, and her results would be automatically separated into categories. She could click on any one of these categories and see all the pages in that category, not just the ones matching her keyword phrase. So, for instance, if she searched for "nirvana," she might see 25 results about the band, followed by 25 results about the religious concept. If she clicked on a link representing the category of the band, she would see all the pages that were about the band, even if some of them did not have the word "nirvana" on the page, e.g. a page about Kurt Cobain, the band's star, that happens not to mention the band by name.

The hierarchical classification of pages in the Open Directory Project forms a tree. Tables of word frequencies that represent nodes in the tree can be built to characterize each node. I build such tables in the following manner.

For each category at the leaves of the tree, I treat all the pages in that category as one large document, visit these pages to collect their text, and stem all words, then compute frequency counts for each stem associated with each category. I then compute the frequency count tables for internal category nodes as the sums of the frequency tables for each of their child nodes, all the way up to the root, in a bottom-

Figure 4.1: Algorithm *Build-Tree-Tables*

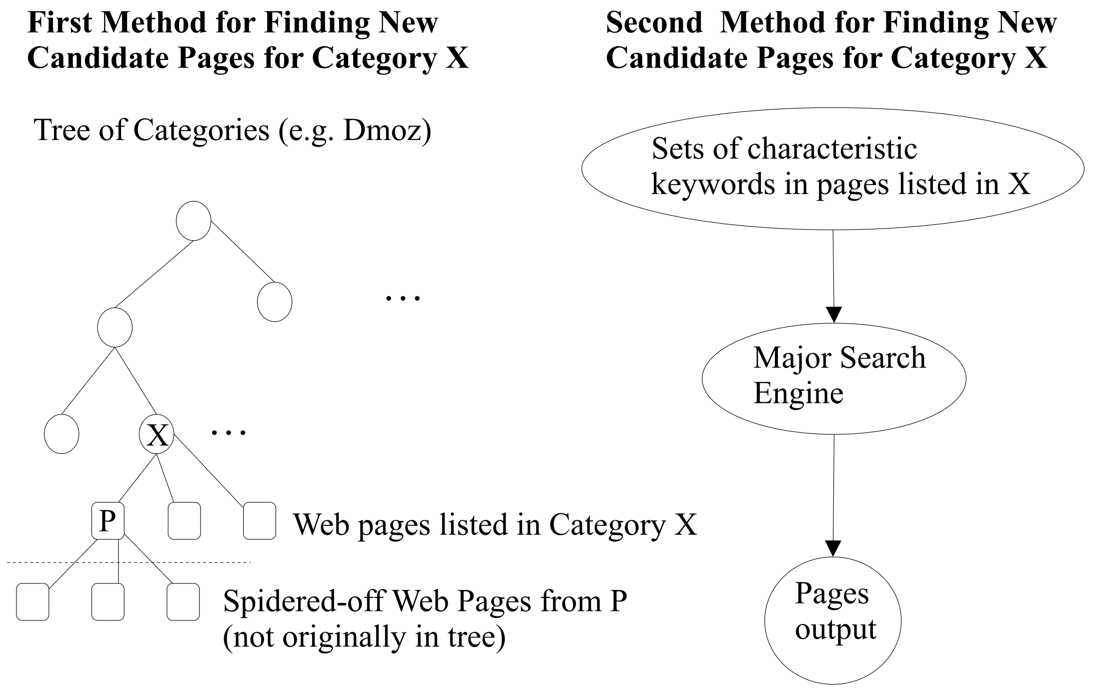
```
1. Call $B2(root)$, where root is a pointer to the root of the tree; return.
2. Subroutine $B2(p)$:
   (a) If $p$ is a leaf node:
       i. Let $d$ be the document formed by concatenating all the documents
          categorized under $p$. Let $freqtable(p, W)$ be the frequency table of
          words in this document, composed of a set of (word,frequency) pairs.
          Compute $freqtable(p)$ by doing a frequency count within $d$, and store
          $freqtable(p)$ in a persistent database.
   (b) if $p$ is not a leaf node:
       i. Let $S$ be the set of all the immediate children of $p$. For each $p_i$ in S,
          call $B2(p_i)$. Compute $freqtable(p)$ by summing up all the frequency
          tables $freqtable(p_i)$ where $i$ is a member of $S$. Frequency tables are
          summed by summing all the frequencies in each table for each word.
```

up fashion. Thus the frequency count table for the root node in the table is the global word stem frequency count. I refer to this algorithm as *Build-Tree-Tables*; it is shown in Figure 4.1.

After these frequency tables have been built, the system can start to grow. Additional pages can come into the system in two main ways. First of all, they can be "spidered" off pages that are already in the system, by following links off of those pages. Secondly, they can come in by doing meta-searches, using other public search engines. Figure 4.2 illustrates this process.

A meta-search represents a quick way to enlarge the number of pages in a particular category. In traditional information retrieval, the TFIDF (term-frequency-inverse-document-frequency) score [5] is used to score terms (words) on their usefulness in separating documents among a collection; that is, as useful documents in doing queries

Figure 4.2: Two Processes for Growing a Category within a Tree that Classifies Web Pages


First Method for Finding New Candidate Pages for Category X
Second Method for Finding New Candidate Pages for Category X
Tree of Categories (e.g. Dmoz)
X
P
Web pages listed in Category X
Spidered-off Web Pages from P (not originally in tree)
Sets of characteristic keywords in pages listed in X
Major Search Engine
Pages output

on that collection. For document d_i in a collection of documents D, and for every term (word stem) w in d_i, if $count(d_i, w)$ is the number of times w appears in d_i, and $countdocs(w)$is the number of documents in which word w appears, let

$$TF(d_i, w) = count(d_i, w) + 1$$

and let

$$IDF(w) = log(\frac{|D| + 1}{countdocs(w)})$$

then the TDIDF score of w in document $\underline{d_i}$ is defined as follows:

$$TFIDF(d_i, w) = \frac{TF(d_i, w)IDF(w)}{\sqrt{\sum_{d_k \in D}(TF(d_k, w)IDF(w))^2}}$$

Starting with the pages in a particular category of interest, a variant of the TFIDF score is computed for each word stem w in that category, which I call the term-frequency-inverse-global frequency (TFIGF). I define the TGIDF score of w with respect to a meta-document created by concatenating all the documents in a category, not with respect to a single document, as in the case of TFIDF. The TFIGF for each word stem is computed with respect to the global word stem frequency; it is the frequency of the term in the pages in the category divided by the frequency of the term in *all* pages in all categories. That is, the denominator, instead of being the log of the number of pages in the document set that the term is present in, is simply the frequency in the global set of pages.

Thus, the TFIGF score reflects the degree to which each word is represented in the pages in the category in question as compared to how frequently it is represented in the universe of pages. So, if C is the category in question, and $countcategory(w, C)$ is the number of times that word w appears in some position in all the documents in C concatenated together, and $countglobal(w)$ is the number of times word w appears in some position in all the documents in all categories concatenated together, then

$$TFIGF(w) = \frac{countcategory(w, C)}{countglobal(w)}$$

Note that $countcategory(w, C)$ and $countglobal(w)$ are computed using Algorithm Build-Tree-Tables (see Figure 4.1). These represent pairs in the tables $freqtable(p)$ where the p are pointers to categories in the tree. The $freqtable(p)$ tables are constucted by that algorithm. The values $countcategory(w, C)$ can be found by looking in $freqtable(p)$ where p is a pointer to C. Similarly, the values $countglobal(w)$ can be found by looking in $freqtable(p)$ where p is a pointer to the root category of the category tree.

Note that TFIDF scores for word stems in order to form document term vectors were devised in order to respond to queries *within* the set of documents; that is, to identify those word stems that were best at separating out documents from one another. This is where the "IDF" part of "TFDIF" comes in, the "inverse document frequency." IDF represents the (log) inverse of the fraction of the documents in the set that contain the word stem. Thus if a particular word stem appears in only a single document, it will be an excellent query term; if it appears in all of them, it will not be good at distinguishing them at all. Thus, we divide by the DF in order to up-weight the terms that appear in few documents so that their dot product scores will be higher, and they will appear higher up on the query output list if a low-frequency term is included in the query.

However, if our purpose is not to do queries within a particular set, but rather to characterize the properties of that set with respect to a universe of documents by using those terms that are characteristic of the set, the TFIGF score makes more sense. The purposes of such a characterization are different than that of query ranking. The characterization is used to gather more pages of the same type, using a search engine or a customized Web agent/spider.

It makes sense that those word stems with the highest TFIGF scores for a particular category are the most characteristic of that category. These word stems can then be passed in conjunction to a search engine to find other pages which are likely to fit into the same category, or in closely related categories.

Such pages can be used in two possible ways. First, they can be passed directly back to the searcher who is looking for more pages about a particular subject than have been pre-classified by the system. These pages are much more likely to be on point than pages that are located using only a single keyword, because the presence of multiple keywords will tend to narrow the focus of the search down to a specific semantic neighborhood.

The second way that such pages can be used is if they are classified into the hierarchy via a multi-resolution classification algorithm. Naive Bayes, or another algorithm, may be used here—Naive Bayes tends to be the most popular algorithm, because it is quite computationally inexpensive and works reasonably well.

4.1 Global versus Local Spidering and Ranking: A Different Way to Look at Search Engines

4.1.1 Design of the “Active Portal” Project

As a test bed for some of the ideas in this thesis, I have built a Web site called the Active Portal project, at www.active-portal.com. The Active Portal project takes its motivation from the observation that the Web tends to be organized around communities, each associated with a set of Web pages, which typically form something similar to a clique in the graph of the Web (although some of the pages are also connected to pages associated with other communities). A similar system for the automated construction of vertical portals is described in [76].

In an initial version of this system, I built Web pages (“portals”) that served approximately 100 communities of Web users; in principle, it could be grown to cover the whole Web, and create a combined search engine/directory as described above. Each one was fairly narrowly defined, for instance, a few of the categories I used were “Economics,” “Cats,” and “Charles Dickens.” I either manually “seeded” each of these portals with pages that I found that fit into the category (100 for each portal) or took pages from a corresponding category in the Open Directory Project, or some combination of these. The seed set became the core set of pages from which spiders seek out new pages. As each new page came into the system, it was tested for membership in the portal.

Initially, the system simply looked for a threshold number of keywords on the candidate page that were characteristic of the other pages in the set, in order to determine whether or not the page should be admitted into the portal. A characteristic word is defined as a word that is over-represented in the pre-classified pages with respect to a background set of random pages, using the TFIGF approach described above. One would expect that pages that have multiple links from pages in the seed set would be more likely to also be a member of the portal, although this factor is not used in determining category membership; Active Portal could alternately use an algorithm such as *Voting-Trumps-NB* as described in Figure 3.7, which would take this into account.

Since the new pages encountered do not affect whether or not their successors are admitted into the portal, a page is admitted to the portal whether it is encountered sooner or later. However, the ranking is affected by the order of encountering pages, as we will see when the ranking technique is described below.

The spidering process continues in iterations; as the portal grows, the system will be following links off of pages that were admitted to the portal earlier. This increases the distance from the initially manually-classified pages, and will increase the error rate, since pages further away from the initial seed set are less likely to be in the category. However, since the criterion for membership in the portal is based on the statistical properties of the initial manually-classified pages, the new pages all need to meet criteria based on those initial pages.

One way to deal this is to explicitly make it harder to make it into the category if you are farther from a manually-classified page; I have not yet done this. Another way is to explicitly strip out links to the pages that are most highly-referenced on the Web as a whole, such as www.yahoo.com and www.microsoft.com, since these are unlikely to fit into a particular category, but the spider would likely visit them; I discuss this in the context of clustering in Section 6.1.

Another way to deal with this is to use user feedback to filter pages, either explicitly in terms of user ratings of pages (admittedly hard to gather) or implicitly in terms of user "click-out" rates (that is, how often visitors click on these pages in the portal). Pages with high ratings or high click-out rates would be moved higher up in the portal. Another strategy would be to allow users to further filter results within the portal with additional query strings.

One way to improve this system, which I may implement in the next version, is to assign a TFIGF score to each word on each page. The total score of a particular page would then be the sum of the scores of the words on that page, divided by the number of words on the page in order to normalize the scores. The advantage of such a system is that it would not be brittle, since the presence of any particular keyword would not be necessary in order to give a page a high rank relative to a category. It would also be possible to gather new pages for a particular category from a reverse-keyword index using different combinations of keywords, even disjoint sets of such keywords.

Thus, unlike a general search engine spider, which wanders the Web in a breadth-first search or some version of a "best-first" search, this spider wanders only off of pages that have already been admitted into the portal. Thus it is much more likely to encounter pages that fit well in the portal. A way to improve this even more is to visit those pages that are referred to by multiple pages in the portal first. Or, a more sophisticated scoring system would give high scores to those pages that had high individual scores and were pointed to by a large number of other pages with high individual scores (recursively, in the manner of Google's PageRank).

This gets to one of the main criticisms I (and others) have had of mainstream search engines. Arguably, Google, of the major search engines, does the best job in ranking the output of a search query (although this is a subjective assessment). But Google's PageRank is relative to the entire Web; it does not rank its output *relative to the query* very well, although it attempts, in some manner (that is not disclosed to the public), to combine the global PageRank value with the query words in producing the final ranking. Other search engines, such as Altavista, rank their pages by some combination of the in-links into each page ranked and the words in the search query, so the rank is somewhat relative to the query. But they make no use of words that are semantically close to words in the search query, which the TFIGF spidering approach described above does. Often searchers do not think of related words that could improve their query results, and the TFIGF approach can automatically gather these.

The approach for page ranking used by Active Portal is twofold. It uses a combination of the number of words that are characteristic of the portal on a page and the number of pages *also in the portal* that point to the page. The number of characteristic words is set via a tuning parameter n[1]. This process is shown in Figure 4.3. Thus the ranking becomes *contextualized* to the particular area of the Web of interest, rather than a global search ranking. (By contextualized, I mean specific to a particular environment, rather than general.) I believe that such a topical, contextualized ranking is more appropriate to most actual searches than the rankings returned by most general search engines, since almost all searches are topical by their nature.

Active Portal, in its present state (because it does not incorporate all of DMOZ, has not done very extensive spidering as compared to the public search engines, and

[1]I have used $n = 500$ for my experiments, but I think this may be too high, in that it introduces too much noise.

Figure 4.3: Active Portal Page Ranking Procedure

1. The characteristic words of a portal are defined as the n (a tuning parameter) top words within the portal ranked in terms of their TFIGF score. To compute these words:

2. Let $topwordstot = 0$; $inlinkstot = 0$.

3. For each page p in the portal:

   (a) Count how many words characteristic to the portal are on page p. Call this $topwords(p)$. Let $topwordstot = topwordstot + totwords(p)$.

   (b) Count how many inlinks p has from other pages. Call this $inlinks(p)$. Let $inlinkstot = inlinkstot + inlinks(p)$.

4. For each page p in the portal, let

$$rank(p) = \frac{inlinks(p)}{inlinkstot} + \frac{topwords(p)}{topwordstot}$$

5. Sort pages with the portal in descending order by $rank(p)$. This treats inlinks and characteristic words equally in terms of giving a page a high rank in the portal.

does not have a reverse-keyword index), is more akin to a directory (like Yahoo! or Dmoz) than a search engine (like Google and AltaVista). In principle, if all these limitations were overcome, it would become a hybrid tool, with the ability to look at results both in terms of categories and in terms of keyword searches, and quickly navigate between each.

4.1.2 An Informal Evaluation of Active Portal

A library science student[2] evaluated Active Portal with respect to four other Internet directories (Yahoo!, LookSmart, Snap, and About) [131]. Two topics were considered that were covered by Active Portal–pages on the television show "Seaquest" and pages on sweatshop labor.

Active Portal had 112 pages listed on Seaquest and 305 on sweatshops. Active Portal had poorer precision than the others (which all had perfect precision, since all of the rest had done their categorization manually), but all the rest had much lower recall (fewer pages listed). This student examined the first 30 results on Active Portal in each list, and most were on topic, or on closely related topics, as judged by her. (Unfortunately, she did not make an actual count of those that were on topic relative to those that were not, so we cannot estimate precision for Active Portal quantitatively here; however I have done so in the context of a comparison to Google in Section 4.1.3 below).

For each topic, Active Portal was compared to three of the others. For Seaquest, Yahoo! had 14 links, LookSmart had 9 links and Snap had 5 links (there was overlap

[2]My sister did this as a class project; obviously, she is not the most unbiased evaluator, but I believe that her results are still valid.

Table 4.1: Number of Pages in Each System for Two Topics

	System				
	Yahoo!	LookSmart	Snap	About	Active Portal
“Seaquest”	14	19	5	N.A.	112
Sweatshops	24	19	N.A.	17	305

between all the directories). For sweatshops, Yahoo! had 24 links, LookSmart had 19 links, and About had 17 links (again, there was overlap; About was substituted for Snap because Snap lacked the sweatshop category). These date are summarized in Table 4.1.

Of course, these results are subject to the limit that they were only examined by one person, albeit someone studying librarianship. All the directories have perfect precision, because we are looking specifically at pages that were manually classified into each category. So, for instance, LookSmart had 100% precision on its pages on Seaquest, because we are only considering the Seaquest category in LookSmart.

Thus, according to this evaluation, Active Portal had much better recall, at the expense of some precision. Generally, there is a tradeoff between recall and precision; trivially, if all documents are returned in response to all information requests, all responses contain all the documents desired, and there is perfect recall, but very low precision. On the other hand, if you shape a very precise query to a search engine, it may return only a single document (which is on topic) when there are several hundred fitting the topic you are looking for; then you have low recall but perfect precision.

Table 4.2: Comparison of Active Portal to Google

Query	Search Engine	Precision (first 100 results)	Total # Results
Nirvana	Active Portal	78	22,316
Nirvana	Google	45	1,070,000
Hanson	Active Portal	53	2,467
Hanson	Google	16	1,620,000

4.1.3 An Informal Comparison of Active Portal to Google

I also compared Active Portal on two topics with queries to Google. The two topics were both about musical groups: Nirvana and Hanson. The queries to Google were these single words: “nirvana” and “hanson.” I manually compared the first 100 results for each. For Nirvana, 78 of the 100 were on topic in Active Portal, as opposed to 45 for Google. For Hanson, 53 were on topic for Active Portal, and 16 for Google. All of Google’s off-topic pages were related to other meanings of these two words (for instance, nirvana’s meaning in Buddhist philosophy). More precise queries would improve Google’s results, but typical users are known to not be particularly skilled at formulating precise queries [75]. Recall is irrelevant, because both systems returned more results than anyone could reasonably look through (for Nirvana, Active Portal had 22,316 links and Google returned about 1,070,000; for Hanson, Active Portal had 2,467 links and Google returned about 1,620,000; Google has a big advantage here because it has done much more spidering). These data are summarized in Table 4.2.

4.1.4 Discussion: Page Ranking and Disambiguation

What is the optimal way to rank pages? The optimal way is in the order that the surfer/searcher would prefer to see them. Unfortunately, given the relatively impoverished nature of many search queries, it is often difficult to gauge the intent of the surfer/searcher. The above approach, by enriching the keywords of the search by finding the semantic neighborhood of those keywords, may clarify this intent to some extent, but not to a perfect extent. However, I have not done the necessary experiments with human subjects that would be needed to compare different ranking systems, such as the one I propose with others publicly available, such as Google's, although I have done some related experiments on page ranking with human subjects in Chapter 5.

An ideal search technique would involve allowing the searcher to ask questions in natural language and would return pages that answered those questions, or ask for information about a topic described in natural language and have such information returned. But the best known of such systems, Ask Jeeves, at www.ask.com, uses case-based natural language processing techniques that were developed in the 1970s, essentially attempting to match each user's queries against a match of manually predefined case-frames. These are akin to scripts or schemas, which were highly popular in the early years of AI, before the resurgence of machine learning and neural networks.

One way to rank pages that are returned as the result of an ambiguous query (and many, if not most, queries are ambiguous) is to separate out the different meanings of the query and then rank the pages within each meaning. In Chapter 6, I discuss techniques that might be used for such disambiguation. All that Google does is show the classification of those pages that have been manually classified by the Open Directory Project, but it does not separate the results from one another. And in the case of an ambiguous query, Google (like other search engines) can perform quite poorly. For instance, if you search for "bear" in Google, it shows you the "Smokey the Bear" main page, which is primarily about forest fires. Presumably, since it ranks high in Google, lots of other other prominent sites link to it, but it is not primarily about bears (although the site does contain some information about bears). It is not until the 12th result, the North American Bear Center, do we find a result that is actually primarily about bears.

In my experience, the search engine that does the best job at disambiguation is Northern Light. Northern Light can, at user request, separate results into what it calls "custom search folders," but it is unclear whether the ranking of the results within each such folder is contextually ranked.[3] In Section 6.1.6, I discuss two new search engines that do something similar.

Some researchers have used the metaphor of "information ecology" to describe the information search process; research groups have been organized around this metaphor, at Xerox PARC most notably (e.g. [90][89]). In this, the searcher is like an animal grazing in its environment, searching for particularly choice niches (parts of the Web) or Web pages within a niche. Each searcher is a different kind of animal, with different tastes in food, but there are also species of searchers, that is, communities of interest. Some people produce "food" that others "eat;" that is, generally, a small subset of the community tends to produce content that is consumed by the rest of the community. Search engines need to take into account human needs

[3]At this writing, Northern Light's public Web search engine is no longer available.

in order to improve their results and their interfaces to those results.

4.2 Using a Tree of Classified Pages to Find More Relevant Pages and Classify Them

Dmoz classifies all of the pages that it contains into a spot in its hierarchy. These pages have been manually classified, which is, of course, very labor-intensive. However, given sufficient pre-classified pages, it should be possible to extend this set in an automated fashion and get reasonable results. Weigend, Wiener, and Pedersen have applied neural networks to classifying text documents hierarchically, where the neural network architecture reflects the hierarchy of the classification [122]. Mladenic has experimented with classifying pages in the Yahoo! hierarchy using the Naive Bayes algorithm with a customized feature set [80].

There are two basic ways (that I have thought of so far) to extend pre-classified hierarchies. The first is to calculate the keyword centroid of all the pages in a particular set, and then pass highly weighted keywords to search engines to get more pages back, which then are classified. This is the approach I have taken in Section 4.3. In this section, I discuss the approach of spidering off existing pages pre-classified in a tree and attempting to classify them.

4.2.1 Hypothesis

A effective method for "growing" a tree of pre-classified Web pages would be by spidering off of pages in this tree and applying a multi-resolution version of the Naive

Bayes algorithm.

4.2.2 Method

For this experiment, I have used all of the categories one or two levels below the "/Science/Biology" category in Dmoz. The tree below "/Science/Biology" is actually deeper and more complex, but for simplicity I have only considered those pages at these three levels (level 0, the root, and levels 1 and 2, respectively one and two levels below it). The three levels of the tree are shown in Tables 4.3, 4.4 and 4.5. A page is considered to be in a category if it is directly classified in that category by Dmoz. That is, in Dmoz's classification scheme (like in Yahoo!'s), not all pages are classified at the leaves of the tree; some are classified at internal ones. In the three level "/Science/Biology" tree here in question, I have included only pages directly classified at the nodes in question, most of which are internal nodes in the full tree.

A hierarchical or multi-resolution version of the Naive Bayes algorithm works as follows. It uses hill-climbing search. It searches the tree, starting at the root node, and considers as categories the root node and all of its immediate children. These become the categories for the first iteration of Naive Bayes. If the root node is the winner, the algorithm stops, and the page being classified is put into that category. If not, the winning child becomes the new root, and the algorithm repeats. It can be applied to as many levels as one desires; in my experiments, I have confined it to a tree of height 2 with 3 levels, the "/Science/Biology" tree discussed above. This algorithm is described in pseudocode in Figure 4.4.

If m is the height of the tree (here 2) and n is the number of children per node (here varying from node to node), then the complexity of this algorithm is $O(kmn)$,

Table 4.3: Dmoz "/Science/Biology" Sub-Categories 1-50 of 174, with the Number of Pages Pre-Classified in each Sub-Category

root category	9	Biotechnology/Stem_Cells	2
Associations	2	Botany	44
Biochemistry	13	Botany/Associations	6
Biochem/Directories	14	Botany/Dendrology	9
Biochem/DNA	9	Botany/Directories	13
Biochem/Education	7	Botany/Education	36
Biochem/Laboratories	9	Botany/Ethnobotany	26
Biochem/Metabolic_Pathways	9	Botany/Journals	37
Biochem/Methods&Techniques	10	Botany/Lichens	16
Biochem/On-line_journals	101	Botany/Paleobotany	20
Biochem/Organizations	9	Botany/Phycology	37
Biochem/Proteins&Enzymes	12	Botany/Plants	16
Biochem/Software	51	Botany/Plant_Pathology	6
Bioinformatics	67	Botany/Plant_Physiology	21
Biophysics	7	Business	26
Biophysics/Internet_Resources	3	Business/Contract_Research_Associations	9
Biophysics/Research_Centers	9	Cell_Biology	12
Biotechnology	45	Cell_Bio/Apoptosis	5
Biotech/Careers	6	Cell_Bio/Cell_Culture	27
Biotech/Companies	121	Cell_Bio/Cell_Cycle	5
Biotech/Courses&Workshops	5	Cell_Bio/Cell_Membrane_and_Adhesion	8
Biotech/Directories	24	Cell_Bio/Cytoskeleton	13
Biotech/Meetings	12	Cell_Bio/Education	8
Biotech/Phytoremediation	4	Cell_Bio/Image_Galleries	5
Biotech/Publications	31	Cell_Bio/Journals	18

Table 4.4: Dmoz "/Science/Biology" Sub-Categories 51-100 of 174, with the Number of Pages Pre-Classified in Each Sub-Category

Cell_Bio/Laboratories	13	Ecology/Publications	1
Cell_Bio/Methods_and_Techniques	4	Ecology/Research_Centers	2
Cell_Bio/Organizations	16	Ecology/Restoration_Ecology	14
Cell_Bio/Signal_Transduction	11	Ecology/Software	19
Cryobiology	31	Ecology/Spatial_Analysis	8
Cryo/Cryonics	33	Ecology/Wildlife_Ecology	212
Cryo/Sperm,_ova,_embryo_storage	25	Education	30
Cryo/Supplies_and_Equipment	6	Education/Bioinformatics	10
Cryo/Umbil_cord_stem_cell_storage	9	Education/Conference&Workshops	3
Developmental_Biology	17	Education/Departments	4
Developmental_Bio/Journals	5	Education/Online_Courses	15
Developmental_Bio/Laboratories	4	Education/Online_Textbooks	10
Developmental_Bio/Model_Organisms	21	Education/Tutorials	38
Developmental_Bio/Organizations	5	Equipment_and_Supplies	11
Directories	33	Equip&Supplies/Contract_Services	3
Ecology	8	Equip&Supplies/Field_Biology	4
Ecology/Aquatic_Ecology	31	Equip&Supplies/Gen_Distributors	12
Ecology/Associations	7	Equip&Supplies/Instrumentation	17
Ecology/Biogeography	11	Equip&Supplies/Living_Systems	7
Ecology/Conferences	5	Equip&Supplies/Microscopy	29
Ecology/Consultants	9	Equip&Supplies/Reagants&Kits	15
Ecology/Ecosystems	10	Equip&Supplies/Software	17
Ecology/Education	9	Evolution	27
Ecology/Microbial_Ecology	4	Evolution/History	9
Ecology/Molecular_Ecology	5	Evolution/Human_Evolution	27

Table 4.5: Dmoz "/Science/Biology" sub-categories 101-174 of 174, with the number of pages pre-classified in each sub-category

Evolution/Internet_Directories	4	Microbio/Taxonomy&Nomenclature	2
Evolution/Journals	1	Microbio/Virology	14
Evolution/Molecular	10	Mycology	21
Evolution/Software	8	Neurobiology	33
Genetics	14	Neurobiology/Addiction	10
Genetics/Employment	0	Neurobiology/Brain_Images	5
Genetics/Eukaryotic	2	Neurobiology/Institutions	10
Genetics/Journals	20	Neurobiology/Journals	4
Genetics/Mutagenicity	0	Neurobiology/Organizations	7
Genetics/Organelles	2	Physiology	14
Genetics/Prokaryotic	6	Reference	15
Genetics/Societal_Issues	22	Research_Centers	12
Genetics/Software	15	Sociobiology	13
Histology	7	Sociobio/Education	5
History	5	Sociobio/Evolutionary_Psychology	11
Immunology	14	Taxonomy	12
Immunology/Associations	9	Taxonomy/Software	8
Immunology/Cytokines	5	Taxonomy/Taxonomy_Map	5
Immunology/Immune_Deficiency_Disorders	18	Theoretical_Biology	10
Immunology/Immunologists	1	Toxicology	26
Immunology/Journals	8	Zoology	26
Immunology/Reference	13	Zoology/Academic_Departments	1
Immunology/Research_Centers	14	Zoology/Acoelomates	18
Immunology/University_Programs	14	Zoology/Animal_Behavior	8
Journals	8	Zoology/Annelida	10
Methods_and_Techniques	4	Zoology/Arthropoda	5
Microbiology	23	Zoology/Chordates	6
Microbiology/Directories	9	Zoology/Echinodermata	9
Microbiology/Environmental_Microbiology	13	Zoology/Images	14
Microbiology/Funding_Sources	1	Zoology/Journals	9
Microbiology/Journals	7	Zoology/Lophophorates	14
Microbiology/Microbiology_Courses	19	Zoology/Misc.-Deuterostomes	4
Microbiology/Microbiology_News	4	Zoology/Misc.-Protostomes	1
Microbiology/Parasites	17	Zoology/Mollusca	11
Microbiology/Protoctista	10	Zoology/Porifera	12
Microbiology/Reference_Labs	1	Zoology/Pseudocoelomates	4
Microbiology/Suppliers_and_Companies	18	Zoology/Radiata	16

Figure 4.4: Multi-Resolution Naive Bayes Algorithm

1. Call MNB*(root,d)* where d is the document to classified and *root* is a pointer to the root of a tree in which it should be classified; exit.
2. Subroutine MNB*(p,d)* where p is a pointer to a node in the tree:
   (a) Form the set P consisting of p and all of its child nodes.
   (b) Consider all the members of P of these as the categories for classification for the Naive Bayes algorithm (see Figure 3.3); note, if a category has not been visited before, its term frequency statistics will need to be computed and stored in a database; otherwise, the statistics can be read from the database.
   (c) Let p_1 be the node representing the category in which Naive Bayes decides to classify d.
   (d) If $p = p_1$, return p as the category in which the algorithm classifies d.
   (e) If $p \neq p_1$, call MNB(p_1, d).

where k is the number of steps in a single Naive Bayes computation. This is not an optimal algorithm; it does not find the best category for each page. An alternative, exhaustive algorithm would be one that considers all of the nodes in the tree as equal and does a single Naive Bayes step for all of them. Let us refer to this as Exhaustive Naive Bayes for the purposes of this section.

However, the exhaustive algorithm visits all the nodes in the tree, rather than just the fraction visited by the hill-climbing search, and therefore is much more costly in time. Since the number of nodes in a tree, however, grows exponentially, while the time for the search is linear in the height of the tree, in most circumstances it is not practical to consider an exhaustive search of all the possible categories that a page could be classified into. However, one compromise might be to use a hill-climbing search as described above to find a candidate category for classification, and then do a exhaustive search in some relatively small neighborhood around that candidate category to see if any other category beats the candidate one in terms of its score on the Naive Bayes algorithm.

One additional way that the multi-resolution version might vary from the normal version of Naive Bayes would be to use all the aggregated statistics of all the descendants of all the children in making the decision between the root and its children at each stage of the algorithm. However, it would not do to do this for the root itself, because it would probably lead to too many pages remaining classified at the root, as an approximately "average" category, and thus defeat the purpose of a tree-like categorization.

One way to test the multi-resolution version of Naive Bayes is to see what proportion of pages are classified back into the category from which they have been pre-classified by the Dmoz editors. However, in the multi-resolution version, a misclassification is not necessarily a complete failure of classification, since there are several different ways in which a particular page might be classified.

For instance, in the three-level "/Science/Biology" tree in question, a page originally classified in a leaf node might be classified in one of the following places: 1) in its original category; 2) in the parent node of its original category; 3) in one of the sibling nodes of its original category; 4) at the root node; or 5) elsewhere in the tree. Classification in (1) is best; of the rest, it seems to me that classification in (2) or (3) is not bad as classification in (5) and perhaps (4) as well.

For a page originally classified at one of the interior nodes; the situation is different. It could be classified in: 1) its original spot; 2) in one of its children; 3) in the root; 4) elsewhere in the tree. Again, (1) and (2) are better classifications than the others. For a page originally classified at the root node, it could be classified by multi-resolution Naive Bayes at the root node or some node below the root. It is not clear that any place below the root is better than any other, although one might argue that the higher up in the tree (closer to the root) the better in this case, because the original classification was minimally specific, so the less specific the classification, the better.

In the original Dmoz 3-level "/Science/Biology" tree, there were 2,767 pages classified by the editors as being in one of the categories on the three levels of the tree. When I actually tried to retrieve these pages from the Web, 224 were not successfully retrieved, and the remaining 2,543 were successfully retrieved. This represents a 91.9 percent successful retrieval rate. The categories in the tree are shown in Tables 4.3, 4.4 and 4.5. As one can see from the tables, there is a very uneven distribution of pages among the categories, which is usual.

In the tree, there are 32 nodes at level 1, and 141 nodes at level 2. This means that there is an average fan-out of 4.4 to level 2, but the distribution of categories at level 2 is very uneven. The structure of the tree means that there are a total of $1 + 32 + 141 = 174$ categories in which each page can be classified. In this example, exhaustively examining each page for each category means that there would be a total of $2,543 * 174$ Naive Bayes steps, or 442,482. However, if you consider the fan-out of 32 at level 1 and the average fan-out of 4.4 at level 2, a multi-resolution Naive Bayes would take an average of $((1 + 32) + (1 + 4.4)) * 2,543 = 38.4 * 2,543 = 97,651.2$ steps. This is obviously much better, and of course the improvement would be even more marked with a bigger tree. And of course this assumes that the complexity of the algorithm is governed by the average fan-out, which is an approximation.

4.2.3 Results and Discussion

Pages Classified in Dmoz Under "/Science/Biology"

The most frequently occurring category assignment at levels 1 or 2 of the tree is "/Science/Biology/Ecology/Wildlife _Ecology," which has 212 pages assigned to it. If we always assigned pages to this category, we would get 212/2767 = 7.7 percent correct. Multi-resolution Naive Bayes does considerably better than this, in this experiment. It classifies 17.9 percent of the pages back into the correct category.[4] This may not seem very impressive, but there are two factors working against the algorithm; the large number of candidate categories, and the close similarity of many of the categories. It is much easier to distinguish pages that are from widely disparate subjects than pages that are within the same general subject area, because the overlap in term usage between pages is likely to be much larger in the latter case than in the former. These factors may contribute to the relatively low percentage classified correctly, although this is only a plausible conjecture at this point.

The most frequently occurring classification, if you consider branches of the tree at level 1 (that is, all pages in a level 1 category or any of its children considered as a single category), is "/Science/Biology/Ecology;" there are 494 pages in that branch. If we always assigned pages to that branch, we would get 494/2767=17.9 percent correct. If you consider whether or not the page has been classified back into the same branch at level 1, the performance is considerably better. Almost half, or 46.7 percent, of pages are classified back into the same level 1 branch. Thus there is a good

[4]Strictly speaking, I should drop each page being classified from the training set and recompute the Naive Bayes probabilities at all nodes without that page in order to test it. This would be cheap computationally if I retain the raw frequencies at all nodes; it would involve a simple substraction.

reason to believe that if pages are being misclassified at level 2, the misclassifications are not severe. Many pages could probably be classified into more than one category, in any case.

It turns out that many of the pages that are misclassified are put into "/Science/Biology/Ecology/Wildlife_Ecology." As we have seen, this is the single most frequent category, so this is not surprising, because Naive Bayes weights the classification by the overall probability that a page falls into a category. If this factor is removed in this case, performance is slightly improved, to 18.2 percent correct at level 2 and 48.2 percent correct at level 1.

Exhaustive Naive Bayes performs somewhat better than multi-resolutional Naive Bayes on these pages, classifying 55.2 percent of them into the correct branch, and 27.2 percent into the correct category.

Pages Linked Off of the /Dmoz "/Science/Biology" Pages

We also consider the performance on pages that are linked off of pages in the "/Science/Biology" hierarchy (at levels one or two as above). The 2,543 pages in "/Science/Biology" that I retrieved from the Web had a total of 68,937 links off of them, or an average of 27.1 links per page (although this is obviously very unevenly distributed). However, due to time and space constraints, I did not want to retrieve all 68,937 pages, so I decided to limit the number retrieved to a maximum of five links per page; this resulted in a list of 9,498 pages, or approximately 3.7 per page (again, unequally distributed). I expected that these linked pages would not classify as well into the categories of the pages that they are linked off as the original pre-classified pages, but they would nevertheless classify into categories at levels 1 and 2 at a rate significantly better than what would be expected by chance.

It turned out that the rate of successful retrieval was considerably less for these linked pages. Of the 9,498 pages on the list of links, only 7,051 were successfully retrieved, for a retrieval rate of about 74 percent. This is not surprising, because manually cataloged addresses tend to be skewed toward home pages and hub and authority pages, which are more likely to have stable Web addresses.

However, assuming that these pages should be classified in the same category as the pages that they were linked from (only a fair-to-middling assumption, but the best that we can make without examining the pages manually), the 7,051 pages classified properly at rates that were higher than what you expect via assignment to the most frequent category (as above). 9.4 percent of the pages classified "correctly" at level 2 (compared to 7.7 percent). 41.7 percent of the pages classified "correctly" at level 1 (that is, in the correct level branch, compared to 17.9 percent). The performance at level 1 was comparable with the original pre-classified pages, which classified in the high 40s. Thus, it appears to be a plausible strategy to use spidering to acquire new pages to fit them into a hierarchy, although the fit will no doubt not be perfect. In combination with a strategy of using clusters of pages to identify sets (see Chapter 6), it could be quite powerful.

On these pages, exhaustive Naive Bayes again did somewhat better, picking the correct branch for 50.6 percent of the pages, and the correct category for 16.2 percent of the pages.

Categories at the Top of the Dmoz Tree

As another experiment in using hierarchical Naive Bayes to classify pages within a tree, I selected some pages from Dmoz where the semantic separation between the categories is subjectively much larger than the sub-categories found within "/Science/Biology." These pages were the 10,381 pages classified at levels 1 and 2 of the entire Dmoz tree; that is, the pages at "/Arts," "/Business," "/Computers," "/Games," "/Health," "/Home," "/News," "/Recreation," "/Reference," "/Regional," "/Science," "/Shopping," and "/Sports."[5] So there were 13 such categories at level 1; below these, there were a total of 467 categories at level 2, for an average fan-out of 35.9. There are fewer categories at level 1 than there were in the case of "/Science/Biology," but the fan-out is much higher, so the total number of categories is higher. Pages were most frequently classified into the branch "/Sports" (2299 pages) and into the specific category "/Shopping/Jewelry" (332 pages). If you pick always the most common branch or category, these correspond to chances of correct classification into the correct branch at level 1 of 2299/10381 = 22.1 percent and chances of 332/10381 = 3.2 percent respectively.

Of the 10,381 pages classified in these categories at levels 1 or 2, the vast majority were classified at level 2 (only 56 pages were classified at level 1). When I tried to retrieve the 10,381 pages from the Web, 8,938 of them returned, or about 86 percent. At level 1, 70.3 percent of these pages classified correctly using multi-resolution Naive Bayes, much higher than the 22.1 percent you would expect by always picking the branch into which pages are most frequently classified. Applying multi-resolution

[5]I did not include the "/Adult," "/Regional," "/World," and "/Bookmarks" parts of the hierarchy because these contained pornographic material, non-English material, and users' bookmarks respectively.

Naive Bayes to both levels, 35.7 percent classified correctly, also much higher than the 3.2 percent that you would expect by always picking the most frequent category. The high level of classification at level 1 means that the severity of the incorrect classifications at level 2 is not great; that is, most of the mis-classified items are being placed in a sister category.

Applying exhaustive Naive Bayes to these pages, performance was somewhat worse. It placed 41.4 percent of pages into the correct branch and 29.0 percent of pages into the correct category.

I spidered off of the 8,938 pages, selecting up to three links per page. This created a list of 20,114 pages. Of these, 18,013 returned from the Web, or about 90 percent. Of course, simply because a page that is pre-classified points to another page, that does not mean that the latter page also falls into the same category. Nevertheless, assuming that such linked pages should be classified into the same category, the correct classification rates are 56.7 percent and 22.3 percent at level 1 (correct branch) and overall respectively, higher than what you would expect by picking the correct branch or specific category. The greater fall off in the classification rates than one would expect with a true pre-classified test set is due to the fact that many of these pages no doubt do not belong to the same category as the page pointing to them. Nevertheless, the relatively high correct classification rates would make extending a directory using links feasible, in my opinion, especially if coupled with collaborative filtering of these links within each directory category ("vertical portal") and a sophisticated link-ordering algorithm within each vertical portal that combines user ratings, category-specific in-links, and degree of membership in the category based on the similarity of the text of the page to the category centroid.

When I applied exhaustive Naive Bayes to these linked pages, I found that the algorithm classified 32.4 percent of the pages into the "correct" branch and 17.9 percent into the "correct category." This was worse than the multi-resolution algorithm.

As an alternative to these two versions of Naive Bayes, a version of the k nearest-neighbor (kNN) algorithm [74] could be used to classify linked pages. A page could be classified into the category that the largest number of its neighbors falls into, using only those neighbors that have been manually-labeled. Here, we would not be using Euclidian distance, but would simply be counting as neighbors those pages that are linked to a particular page (in either direction).

4.2.4 Summary of the Performance of Multi-Resolution Naive Bayes

A summary of the performance of multi-resolution Naive Bayes with respect to the four sets of pages on which it has been tested is given in Table 4.6. As we have seen, the performance is better when there is more semantic separation between the categories. Performance is uniformly worse on the linked pages (second and fourth sets in the table) than it is on the pre-classified ones, because of the assumption that they belong to the same category as the pages from which they are linked, which is only a fair assumption. Multi-resolution Naive Bayes performs better than exhaustive Naive Bayes on the root pages and on the pages linked from them, and worse than exhaustive Naive Bayes on the Dmoz "/Science/Biology" pages and the pages linked from them. It is not clear precisely what circumstances lead one of these to be perform better than the other; further experiments are needed. However, it is

Table 4.6: Performance of Multi-Resolution (M) Naive Bayes and Exhaustive (X) Naive Bayes on Four Sets of Pages

Set of Pages	categories	pages	Alg.	% Correct Branch	% Correct Category
Dmoz "/Science/Biology"	174	2,543	M	46.7	17.9
			X	55.2	27.2
Linked off Dmoz "/Science/Biology"	174	7,051	M	41.7	9.4
			X	50.6	16.2
Root Dmoz Categories	378	8,938	M	70.3	35.7
			X	41.4	29.0
Linked Off of Root Dmoz Categories	378	18,013	M	56.7	22.3
			X	32.4	17.9

a perhaps surprising fact that multi-resolution Naive Bayes can sometimes perform better. This may be because it can sometimes do a better job of selecting the right branch and filtering out noisy classifications based on too little data at the level of the individual category.

4.3 Extending Pre-Classified Sets by Using their Centroids and Reverse-Keyword Search Engines

4.3.1 Hypothesis

A TFIDF centroid of a set of pre-classified pages can be useful in finding words to pass to search engines to find additional pages in the same class (category).

4.3.2 Method

Yahoo! and Dmoz both provide large numbers of pre-classified pages. One approach is to compute the TFIDF centroid of one of the categories in such a classification scheme. Here, each page is represented by a normalized TFIDF vector in the usual vector space model [5], using a root-sum-squared method for normalization so each page is treated equally in the centroid. The centroid is the normalized average of all of these vectors, for all of the pages in the category. This method of computing the centroid is given in pseudocode in Figure 4.5.

Centroids are usually used in the context of clustering, for example in [56], but in this case we are not using clusters, which are typically created through automatic partitioning within sets of documents without supervision by a user or users. Here, we are simply using the centroid to characterize a set of documents, which have already been clustered.

Figure 4.5: Computing the Normalized TFIDF Centroid of a Set of Documents D

1. For each document d_i in D:

   (a) For each wordstem w in d_i, if $count(d_i, w)$ is the number of times w appears in d_i, and $countdocs(w)$ is the number of documents in D in which word w appears, let
   $$TF(d_i, w) = count(d_i, w) + 1$$
   and let
   $$IDF(w) = log(\frac{|D| + 1}{countdocs(w)})$$
   then the TDIDF score of w in document $\underline{d_i}$ is defined as follows:
   $$TFIDF(d_i, w) = \frac{TF(d_i, w)IDF(w)}{\sqrt{\sum_{d_k \in D}(TF(d_k, w)IDF(w))^2}}$$

   (b) Represent d_i by a vector of the $TFIDF(d_i, w)$ for all w in d_i.

2. Let $c = \sum_i d_i$ be the non-normalized centroid vector of the d_i, where the d_i are in their (normalized) vector representations.

3. For all components c_j of c, let $rss(c) = \sqrt{\sum_j (c_j)^2}$. Redefine c, the normalized centroid vector, as $c \Leftarrow \frac{c}{rss(c)}$, and return it.

4.3.3 Results of TFIDF Centroid Computation: Physics

Table 4.7 shows the centroid that was computed using 1,782 of the 1,899 pages listed in Dmoz's "/Science/Physics" category (which has a number of sub-categories that are aggregated here). (Some of the pages could not be retrieved, because of dead links.) There are actually about 50,000 word stems listed in the centroid, with the sum squares of their scores equaling one. (Stemming was performed on each document's representation before the TFIDF scores were computed, so the centroid vector contains only the word stems as well.) I list only the top 50 word stems here, sorted by descending value in the centroid's vector representation, after dropping those words found on a "stop list" of common words such as "the," which was in part obtained from a colleague.

Many of the above words are closely associated with physics, but some of them are only associated with universities and not physics in particular (e.g. "department," "research," "lecture," "course," and "student"). Some of them are very general words (e.g. "main," "please," "time," "light," etc.).[6] In my judgment, the following 13 words in the list in Table 4.7 are tightly associated with physics: "physics," "quantum," "plasma," "particle," "neutrino," "einstein," "laser," "vacuum," "fusion," "nuclear," "mass," "mechanics,"and "wave." I did a Google search using the first ten terms in the table, which returned 615 results, of which I considered the first 264.[7]

[6]The word "requested" appears because it is part of pages that are no longer present, and the text "the requested URL is no longer present on the server" is returned. (I did not attempt to remove these, although I probably should have. The trouble is that error messages returned by web servers can vary considerably, so automatically detecting such errors is a bit of a task. Another error commonly returned is due to the fact that the web servers can detect that the automatic page retrieval software I am using is not interactive and therefore does not support frames. However, such errors only introduce a small amount of noise into the centroid. Iterations of centroid generation like this can be used to increase the size of "stop lists", that is, words that are ignored by the system.)

[7]The batch retrieval program that I used, which is the WWW::Search Perl module from the

Table 4.7: Centroid of Word Stems Derived from Dmoz's "/Science/Physics" Category

rank	word stem	score	rank	word stem	score	rank	word stem	score
1	physic	0.20	11	research	0.09	21	neutrino	0.07
2	main	0.14	12	energy	0.09	22	support	0.07
3	quantum	0.11	13	department	0.09	23	black	0.07
4	plasma	0.11	14	information	0.09	24	einstein	0.07
5	theory	0.11	15	time	0.08	25	laser	0.07
6	particle	0.10	16	light	0.08	26	vacuum	0.07
7	science	0.10	17	experiment	0.08	27	fusion	0.07
8	please	0.10	18	group	0.07	28	institute	0.07
9	university	0.10	19	field	0.07	29	sec	0.07
10	search	0.09	20	hole	0.07	30	equation	0.06

rank	word stem	score	rank	word stem	score
31	space	0.06	41	moved	0.06
32	lecture	0.06	42	object	0.06
33	course	0.06	43	problem	0.06
34	nuclear	0.07	44	list	0.06
35	high	0.06	45	port	0.06
36	mass	0.06	46	journal	0.06
37	student	0.06	47	index	0.06
38	mechanic	0.06	48	menu	0.06
39	data	0.06	49	wave	0.06
40	contact	0.06	50	service	0.06

I manually examined the first 30 of the Google physics results and found that they were all either entirely or partially about physics and/or closely related subjects (e.g. astronomy, astrophysics). A few were pages put together by libraries that listed physics resources along with other resources, usually about other sciences, and some of the pages were out of the mainstream, offering alternative theories about physics.

The idea of this methodology is to use the centroid to gather additional materials about a topic, so it is interesting to see how many of the 264 results were found in the original set of 1,899 pages. It turns out that not a single one of them were in the original set. In a way, this is a good thing, in that it shows that this technique can use the properties of one set of pages to find a set of similar pages. Also, if you think about it, there are thousands of pages about physics on the Web, and the Dmoz editors have only gathered a small sample of them, so it is not likely that given a page at random, it will be in the Dmoz set.

4.3.4 Results of TFIDF Centroid Computation: Chemistry

There were 1,472 Dmoz pages in the chemistry category, of which 1,379 were successfully retrieved from the Web. These latter pages were used to compute the centroid. Table 4.8 shows the top 50 word stems in this chemistry centroid, after removing stop words. In my judgment, the following 7 words corresponding to word stems from the list in Table 4.8 are specific to chemistry: "nmr," "chemistry," "resonance," "corrosion," "molecular," "structure," and "compound."

Comprehensive Perl Archive Network (CPAN), was only able to dump the first 264 of the 615 results into a file of URLs, which probably had to do with the responsiveness of Google to batch queries on the day in question, but did not matter, because 264 results are certainly enough for the purpose of compiling statistics.

Table 4.8: Word Stems in the Chemistry Centroid Derived from the Dmoz "/Science/Chemistry" Category

rank	word stem	score	rank	word stem	score	rank	word stem	score
1	nmr	0.23	11	product	0.10	21	instrument	0.07
2	chemistri	0.20	12	inform	0.10	22	reson	0.07
3	main	0.15	13	scienc	0.10	23	student	0.07
4	support	0.13	14	research	0.09	24	resourc	0.07
5	chemic	0.13	15	informat	0.09	25	data	0.07
6	sec	0.13	16	softwar	0.08	26	corros	0.07
7	search	0.12	17	depart	0.08	27	group	0.06
8	magnet	0.11	18	byte	0.08	28	outdat	0.06
9	servic	0.11	19	laboratori	0.08	29	compani	0.06
10	univers	0.10	20	contact	0.07	30	mail	0.06

rank	word stem	score	rank	word stem	score
31	databas	0.06	41	menu	0.06
32	design	0.06	42	updat	0.06
33	copyright	0.06	43	field	0.06
34	molecular	0.06	44	compound	0.06
35	analysi	0.06	45	exist	0.06
36	view	0.06	46	object	0.06
37	list	0.06	47	nav	0.06
38	move	0.06	48	bottom	0.06
39	structur	0.06	49	index	0.06
40	port	0.06	50	faculti	0.06

I passed the top ten of the words corresponding to the top ten word stems to Google. (None of these word stems were on the stop list). This gave 40 results. I examined these results manually, and all of them are about chemistry, in total or in part, or closely related fields. Because of the heavy use of techniques from physics in chemistry, specifically NMR (nuclear magnetic resonance), the chemistry pages are more likely to have content about physics than vice versa, it appears.

4.3.5 Results of Classifying Search Engine Output with Naive Bayes

When I use only the top five words in the list in Table 4.8 as a query to Google, this increases the number of results to over 7,000, and again, they appear to be mainly about chemistry. I used the batch program to pull the top 500 URLs from this list. If I do the same thing for the physics keywords, I get over 9,000 results, again apparently mainly about physics. I used the batch program to pull the top 500 from this list as well. This gives us two sets of the same size that we can compare with Naive Bayes to see how well they fall into the two sets.

Unlike the other experiments on the chemistry and physics data sets described in this thesis, in which Naive Bayes and variant algorithms were used to see how well pages were classified into sub-categories *within* the categories of chemistry and physics, here we simply want to use the two relatively coarse-grained categories of chemistry and physics as our two categories for classification, and compare classification scores for three groups of pages: (i) pages in training sets of chemistry and physics pages drawn from the Dmoz sample, (ii) pages in test sets of the chemistry and physics pages from that sample (what is left over after removing the training sets), and (iii) the additional pages that were gathered from the Web. Here, we are developing a *single Naive Bayes classifier* to discriminate between chemistry and physics pages.

For the Dmoz pages, I selected 75 percent of the pages at random in each case for each training set, and the remaining 25 percent for each test set. Since there were 1,375 pages in the chemistry set (after ignoring pages that were not retrieved from the Web or contained no text other than stop words), this meant there were 1,108 pages in the training set and 347 in the test set. Similarly, for the physics pages, since there were 1,799 pages, there were 1,337 pages in the training set and 442 in the test set. Overall, therefore, the training set for the two-topic classifier therefore contains 1,108+1,337=2,445 pages, and the training set contains 347+442=789 pages.

Of the 500 results that Google found from the top-five word stems for each of the chemistry and physics centroids, I was able to actually retrieve from the Web 446 pages for chemistry that contained meaningful text (other than stop words) and 449 pages for physics.

I used Naive Bayes to train a single classifier that discriminates between pages in the Dmoz physics and chemistry sets. I also tested this classifier on the pages that were returned by Google for physics and for chemistry using the top-five words in each respective centroid as a query string. This procedure is given in pseudocode in Figure 4.6. The performance on Naive Bayes for this classifier for the training and test sets from Dmoz and the test set from Google are summarized in Table 4.9.

What is fascinating here is that the performance on the Google test pages is actually better than that on the Dmoz test pages. Using the top five words is somewhat

Figure 4.6: Discriminating between Two Topics using a Naive Bayes Classifier, and Using this Classifier to Classify Search Engine Output using Top Words from a TFIDF Centroid for Each Topic

1. Divide a set of pages from Dmoz on a particular topic (e.g. Physics) into training and test sets.
2. Do the same for a set of pages from Dmoz on a second topic (e.g. Chemistry).
3. Create a training set which is the union of the training sets from (1) and (2) above; similarly for the test set.
4. Create a two-class Naive Bayes classifier for these two topics using this training set, which classifies pages into either one of these classes.
5. Report performance for this classifier on the 4 sets (2 training, 2 test) in (1) and (2).
6. For the first topic (using the pages in the training set), compute the normalized TFIDF centroid (as described in Figure 4.5). Do the same for the second topic.
7. For the first topic, retrieve a set of n pages from a search engine (e.g. Google) using the top-five words from the normalized TFIDF centroid. Do the same for the second centroid.
8. Use the two-class Naive Bayes classifier to see how well these pages returned by the search engine are classified back into the correct one of the two categories. (Here, we assume that each set of search engine results accurately belongs with the respective topic.) Report performance statistics for each set of n pages retrieved in (7) above.

Table 4.9: Naive Bayes Performance in Discriminating between Chemistry and Physics on Dmoz Training and Test Sets and on Google Results

		size	number correct	percent correct
all pages	Dmoz training set	2365	2111	89.3
	Dmoz test set	789	602	76.3
	Google test set	895	783	87.5
physics pages only	Dmoz training set	1337	1119	83.7
	Dmoz test set	442	289	65.4
	Google test set	449	363	80.8
chemistry pages only	Dmoz training set	1028	992	96.5
	Dmoz test set	347	313	90.2
	Google test set	446	420	94.2

similar to using a decision tree algorithm in which the words that are best at discriminating a set from the background or from other sets, and apparently, at least in this case, this is a very effective way of finding additional pages that are on topic, at least in the "opinion" of the Naive Bayes algorithm. Of course, as I have noted, the mathematics of Naive Bayes and of computing the centroid are similar, so it should not be that much of a shock—what I am doing here is somewhat circular. Nevertheless, it would be useful in practice, as a meta-search-based methodology for locating additional pages about a particular topic, especially if one is not planning on building a reverse-keyword index search engine of one's own with full coverage of the Web and therefore cannot classify pages as they come into that search engine.

4.3.6 Hypothesis

A TFIGF centroid of a set of pre-classified pages can be useful in finding words to pass to search engines to find additional pages in the same class (category).

4.3.7 Method

I also tried the TFIGF method described at the beginning of this chapter to compute the centroid. If the word stem does not appear in the global corpus at all, I assign it a global frequency of 0.5, so the TFIGF value does not become undefined. These words are the most characteristic of all (although there is sometimes a problem with misspelled words or odd neologisms.) The vector is then normalized in the usual manner, and the centroid is computed from all of the TFIGF vectors. This process is detailed in Figure 4.7. I have performed this process for the chemistry and physics sets we have been discussing.

4.3.8 Results of TFIGF Centroid Computation: Physics

The results for the physics set are given in Table 4.10.

In my judgment, the following 20 words corresponding to word stems in the TFIGF centroid shown in Table 4.10 are tightly related to physics: "electric," "atomic," "relativistic," "dirac," "physics," "tokamak," "oscillation," "schwarzschild," "boson," "spacetime," "pira," "aps," "iop," "lorentz," "superstring," "qcd," "positron," "hamiltonian," "baryon," and "muon." ("APS" stands for American Physical Society, "PIRA" for Physics Instructional Research Association, and "IOP" for the Institute of Physics: the other words are terms commonly understood by physicists). This is somewhat more than the 13 words that I found for the TFIDF centroid, but of course this is a subjective assessment. However, this gives some evidence, in my opinion, that the TFIGF method is better at identifying words specific to a topic.

Figure 4.7: Computing the Normalized TFIGF Centroid of a Set of Documents D

1. For each wordstem w that appears somewhere in D, let $countglobal(w)$ be the number of times that w appears in all the positions in a global corpus (this corpus consists of a large sample of random pages from Yahoo!). If w is not present in the global corpus, set $countglobal(w) = 0.5$.

2. For each document d_i in D:

   (a) For each wordstem w in d_i, let $countdoc(w)$ be the number of times w appears in D. Let

   $$TFIGF_U(w) = \frac{countdoc(w)}{countglobal(w)}$$

3. (normalization step) For each document d_i in D,

   (a) For each wordstem w in d_i and for each wordstem w_k in d_i let

   $$TFIGF_N(w) = \frac{TFIGF_U(w)}{\sqrt{\sum_k (TDIGF_u(w_k))^2}}$$

   (b) Represent d_i by a vector consisting of the $TFIGF_N(w)$ for all w in d_i.

4. Let $c = \sum_i d_i$ be the non-normalized centroid vector of the d_i, where the d_i are in their (normalized) vector representations.

5. For all components c_j of c, let $rss(c) = \sqrt{\sum_j (c_j)^2}$. Redefine c, the normalized centroid vector, as $c \Leftarrow \frac{c}{rss(c)}$, and return it.

Table 4.10: Centroid Computed from Dmoz Physics TFIGF Vectors

rank	word stem	score	rank	word stem	score	rank	word stem	score
1	educat	0.30	11	assocati	0.13	21	oscillat	0.07
2	online	0.25	12	office	0.12	22	official	0.07
3	engineer	0.23	13	arxiv	0.11	23	optical	0.07
4	electric	0.21	14	issue	0.11	24	employment	0.07
5	update	0.19	15	dirac	0.13	25	unite	0.06
6	internat	0.18	16	october	0.09	26	main	0.06
7	atomic	0.17	17	physics	0.08	27	pira	0.06
8	relativist	0.16	18	activiti	0.08	28	schwarzschild	0.06
9	equation	0.15	19	tokamak	0.07	29	boson	0.06
10	archive	0.14	20	adminarxiv	0.07	30	spacetim	0.07

rank	word stem	score	rank	word stem	score
31	phy	0.06	41	qcd	0.05
32	aps	0.05	42	equipment	0.05
33	iop	0.05	43	positron	0.05
34	lorentz	0.06	44	hamiltonian	0.05
35	internal	0.05	45	schrdinger	0.05
36	persist	0.05	46	observat	0.05
37	superstr	0.05	47	muon	0.04
38	annoucement	0.05	48	weizmann	0.04
39	explorer	0.05	49	baryon	0.04
40	explore	0.05	50	science	0.04

4.3.9 Results of TFIGF Centroid Computation: Chemistry

The centroid computed for the chemistry pages using the TFIGF method is given in Table 4.11.

Table 4.11: Centroid Computed from Dmoz Chemistry TFIGF Vectors

rank	word stem	score	rank	word stem	score	rank	word stem	score
1	educat	0.22	11	official	0.12	21	sec	0.08
2	update	0.28	12	equipment	0.12	22	unite	0.07
3	nmr	0.21	13	acs	0.11	23	fid	0.07
4	online	0.20	14	informat	0.10	24	outdat	0.07
5	engineer	0.19	15	archive	0.10	25	chemistry	0.07
6	internat	0.16	16	employment	0.10	26	main	0.07
7	electrochem	0.16	17	atomic	0.10	27	availabl	0.06
8	organic	0.15	18	october	0.09	28	admission	0.06
9	office	0.13	19	explorer	0.09	29	unavail	0.06
10	associat	0.13	20	inaccur	0.08	30	nav	0.07

rank	word stem	score	rank	word stem	score
31	chemi	0.06	41	bottom	0.05
32	include	0.06	42	peptid	0.05
33	decoupl	0.06	43	chemistrycoach	0.05
34	potentiostat	0.06	44	bruker	0.05
35	banner	0.05	45	issue	0.05
36	activiti	0.05	46	cas	0.05
37	temporarili	0.05	47	chiral	0.05
38	ordere	0.05	48	chromaticographic	0.05
39	persist	0.05	49	chemweb	0.04
40	announcement	0.05	50	correctli	0.04

In my judgment, the following 12 words corresponding to the word stems in Table 4.11 are directly about chemistry: “nmr,” “electrochemical,” “organic,” “acs,” “fid,” “chemistry,” “potentiostat,” “peptide,” “bruker,” “cas,” “chiral,” and “chromatographic." "ACS" stands for the American Chemical Society; "FID" is a term relating to NMR, "CAS" is the Chemical Abstracts Service, and Bruker is a manufacturer of scientific equipment, including NMR machines. These 12 words are a larger set than those that I judged to be directly about chemistry from the TFIDF top 50 word stems, but again, this judgment is subjective.

4.3.10 Hypothesis

A list that combines wordstems from both the TFIDF and TFIGF would contain more topical keywords than either the TFIDF or TFIGF centroids alone.

4.3.11 Method

The best performance, I found, was obtained by combining the TFIDF and the TFIGF results in the following manner. For each word stem, its ranks on each of the TFIDF and TFIGF centroid vector lists, sorted by TFIDF or TFIGF score in a descending manner, were summed. If a particular word stem was not found on one of the lists, it was considered to be at the end of the list for purposes of computing its rank on that list. The details of this process are given in Figure 4.8. Doing this significantly boosted the number of words specific to the field found in the top 50, in my judgment.

4.3.12 Results

The result for the physics set is shown in Table 4.12. Scores are omitted because of the way this computation was done.

Figure 4.8: Combining TFIDF and TFIGF Results for a Set of Documents D

1. Let $c_{TFIDF}(D)$ be the normalized TDIDF centroid for a set of documents. Let n_{TFIDF} be the number of words present in this centroid.

2. Let $c_{TFIGF}(D)$ be the normalized TDIGF centroid for a set of documents. Let n_{TFIGF} be the number of words present in this centroid.

3. Let W be the set of wordstems present in either or both centroids. For each w_i in W:

   (a) Let $r_{TFIDF}(w_i)$ be the rank of w_i in $c_{TFIDF}(D)$; that is, its rank when all the values in the vector representing the centroid are sorted in descending order of magnitude; if $w_i$is not present in $c_{TFIDF}(D)$, then let $r_{TFIDF}(w_i) = (n_{TFIDF} + 1)$.

   (b) Let $r_{TFIGF}(w_i)$ be the rank of w_i in $c_{TFIGF}(D)$; that is, its rank when all the values in the vector representing the centroid are sorted in descending order of magnitude; if $w_i$is not present in $c_{TFIDF}(D)$, then let $r_{TFIGF}(w_i) = (n_{TFIGF} + 1)$.

   (c) Let $r(w_i) = r_{TFIDF}(w_i) + r_{TFIGF}(w_i)$.

4. Order the wordstems in D based on ascending values of $r(w)$.

Table 4.12: Combined Physics Centroid

rank	word stem	rank	word stem	rank	word stem
1	sec	11	logo	21	tesla
2	quantum	12	schwarzschild	22	planck
3	einstein	13	pira	23	ph
4	left	14	lorentz	24	neutron
5	cern	15	synchrotron	25	beam
6	dirac	16	adminarxiv	26	weizmann
7	vacuum	17	byte	27	banner
8	hep	18	muon	28	header
9	tokamak	19	doit	29	octob
10	fermilab	20	qcd	30	hepic

rank	word stem	rank	word stem
31	unabl	41	pppl
32	iop	42	collis
33	proton	43	toolbar
34	bohm	44	bohr
35	bottom	45	cofficici
36	lagrangian	46	wavelength
37	quant	47	heisenberg
38	hamiltonian	48	nobel
39	kek	49	physicsweb
40	casimir	50	schrdinger

In my judgment, the following 29 words corresponding to the word stems in Table 4.12 are directly about physics: “quantum,” “einstein,” “cern,” “dirac,” “vacuum,” “hep,” “tokamak,” “fermilab,” “schwarzschild,” “pira,” “lorentz,” “synchrotron,” “muon,” “qcd,” “tesla,” “planck,” “neutron,” “hepic,” “iop,” “proton,” “bohm,” “lagrangian,” “hamiltonian,” “kek,” “casimir,” “pppl,” “bohr,” “wavelength,” and “heisenberg.” This is considerably more than was found by the TFIDF or TFIGF methods alone, which found 13 and 20 words respectively, and indicates that a balance between up-weighting rare words and up-weighting words that appear in relatively few of the documents in the corpus may be a fruitful approach.

Table 4.13 shows the top 50 combined-rank TFIDF-TFIGF centroid word stems for the chemistry pages. In my judgment, the following 17 words corresponding to word stems in the list in Table 4.13 are directly about chemistry: “nmr,” “bruker,” “spectra,” “potentiostat,” “seaborg,” “varian,” “hornak,” “spectral,” “nqr,” “chemistry,” “nuclei,” “deuterium,” “tecmap,” “fid,” and “bmrl.”[8] This is somewhat more than the 12 found by the TFIGF method alone, and significantly more than the 7 found by the TFIDF method.

Note that the goal of this section of this chapter has been to identify keywords that are salient to a particular topic, given a corpus of text (here, preclassified pages from Dmoz) that is on a particular topic (here, chemistry or physics). TFIDF and TFIGF centroids, and a combined TFIDF/TFIGF technique computed from these centroids, have been used to identify the salient keywords in this section. This is different from the goal of Naive Bayes, which is (in the textual context) to classify texts into

[8]Some of the word stems in table 4.13 correspond to Web-related words coming from error messages generated by Web servers; it is difficult to weed out all of these messages using stop lists; the words “outdated,” “unavailable,” “temporarily,” and “correctly” are mainly from these messages.

Table 4.13: Combined Chemistry Centroid

rank	word stem	rank	word stem	rank	word stem
1	nmr	11	spectra	21	doit
2	sec	12	menu	22	chemistrycoach
3	informat	13	educat	23	byte
4	outdat	14	unavail	24	potentiostat
5	nav	15	support	25	seaborg
6	bottom	16	temporarili	26	associat
7	bruker	17	correctli	27	spell
8	banner	18	internat	28	persist
9	header	19	exist	29	nobel
10	header	20	left	30	varian

rank	word stem	rank	word stem
31	chemweb	41	chemistri
32	ucsf	42	nuclei
33	chemsoc	43	deuter
34	octob	44	maintain
35	availabl	45	tecmag
36	bhome	46	tripo
37	bodi	47	fid
38	hornak	48	bookmark
39	spectral	49	bmrl
40	nqr	50	sidebar

categories based on other, pre-classified pages that have already been manually placed in those categories.

However, the mathematics of the TFIGF terms that have been used in this section is somewhat similar to that of the individual probabilities that are used in Naive Bayes. The difference is that in TFIGF the terms represent the ratio between the frequency of a given word in a particular document and the global frequency of that word (see Figure 4.7), whereas in Naive Bayes the terms represent the ratio between the frequency of a given word in a document that is the concatentation of all documents classified in that category and the sum of the total number of word positions in that concatenation and the global vocabulary (see Figure 3.3).

4.4 Review of the Contributions of this Chapter

The first contribution of this chapter, illustrated by the design of the Active Portal system described in Section 4.1, is mainly a conceptual one. I point out that search results always should be ranked in terms of the needs of the searcher. A list of globally-ranked results that contains a collection of pages on semantically unrelated topics is too noisy, if the searcher is only looking for pages on one of these topics. I therefore design a mechanism by which pages could be ranked *within* each particular topic; such a mechanism requires tuning and possible redesign based on user feedback (which I have not yet collected), but I argue that it should combine both the use of words over-represented within the category (via the TFIGF score) and a count of in-links to each page from other pages *within* the category (its "authority"-like nature). In addition, one could add a count of out-links, to measure the "hub"-like nature of the page–I have not done this yet. In addition, these factors could be combined in a recursive manner, like Google does, but again this is a direction for further research.

The experiments I have conducted with Active Portal indicate that spidering off of pre-classified pages is an effective strategy for locating pages on the same topic; this is consistant with the results of Chapter 3. This effectiveness was measured by comparing the results of Active Portal on some topics with some other directory systems, and with Google.

As an additional experiment with "growing" a tree of pre-classified pages, I employed a novel multi-resolution version of the the Naive Bayes algorithm, which is more scalable than Naive Bayes when applied to a large tree of categories. One significant contribution was the finding that such an algorithm was effective, compared to assignment to the most frequent category, in assigning pages back to the correct branch of the tree, and to a lesser extent, to the same precise category. Another significant result/contribution here was the finding that the greater initial semantic separation of the categories led to better performance; thus I found that the algorithm did a better job separating pages taken from the top-level Dmoz categories, which are widely divergent in topic, as opposed to the closely-related pages within the "/Science/Biology" tree. I also found that pages spidered off of pages in these categories were also classified back into the categories at a rate higher than what you would expect if you just assigned pages to the most frequent category.

The final main contribution of this chapter is the idea of mining highly salient keywords from the TFIDF or TFIGF centroids of sets of pages and passing sets of those keywords to search engines in order to identify additional pages in the category. I demonstrate that such a method is good at identifying salient keywords—a method involving a combination of TFIDF and TFIGF appears best of all—and the results returned by Google to searches using these keywords result in a large proportion of pages that belong to the category, according to Naive Bayes. This would indicate that this is an effective method of finding more pages on a particular topic.

Chapter 5

Using Feedback to Find Better Results

5.1 Models of Web Page Quality and Relevance

In the information retrieval literature, relevance is usually thought of as a Boolean value—either the result is relevant, or it is not. In classic studies of relevance feedback, a set of results to a query are presented to the user, and the user marks some of these as relevant or non-relevant. Terms drawn from those documents tagged as relevant are used to refine the query. This has been shown to improve the precision of retrieval (that is, the percentage of documents returned that are relevant), e.g. in [98, 107].

Precision and recall are useful concepts when you have a fixed number of documents that fit into any category. In that case, as is conventional in the literature, precision is defined as a the percentage of the retrieved documents that fall into the desired category (that is, are relevant to a particular information need, as defined by the user or a predefined expert assignment), and recall is defined as the percentage of the relevant documents that have been retrieved. In the Web context, where the number of documents in any particular category tends to grow monotonically over time, achieving high precision (so as to have a low signal-to-noise ratio for the user) and finding a large number of highly relevant documents, and ranking them well, is more important than having high overall recall, since most users will not be able to process any but the most highly relevant documents. Thus precision usually becomes more important than recall, since there is usually a flood of information on any topic, more than any one person can process. Recall is important in that you want to make sure that the most relevant documents are included in the set. The distribution of relevance judgments and the theory of such judgments has been examined in [8].

5.1.1 Hypothesis

The presence of TFIGF keywords in and/or a count of in-links to a document can predict whether or not that document receives high quality or relevance (to a particular topic) ratings from a pool of subjects.

5.1.2 Method

In the interest of discovering what features contribute to high levels of precision and high mean relevance of returned documents, I conducted the following experiment. I used a set of 500 documents which I manually-selected, all on public policy topics related to Wisconsin, 100 each on the following topics: Wisconsin economy, Wisconsin education, Wisconsin environment, Wisconsin government and politics, and

Wisconsin health care. (These were the same pages used in the experiment reported in Section 3.1.)

A group of 107 student-subjects were used in the experiment, all students of Prof. Jo Ann Oravec at the University of Wisconsin at Whitewater. Each student was assigned to one of the categories, and was asked to rate each page in that category along two dimensions, relevance and quality, each scaled between 1 and 10. The students were instructed that relevance simply meant to what extent the page fit into the category in question, and quality meant the overall quality of the information on the page, independent of the relevance of the page to the category. (So, for instance, if one of the pages in the Wisconsin education set actually contained very high quality information on, say, California wineries, it would get a relevance value of 1 and a quality value of 10.)

The subjects were each assigned to make 100 relevance and 100 quality judgments of the 100 pages in the set to which they were assigned. In aggregate, the subjects made 8,982 relevance judgments and 9,013 quality judgments, indicating that they actually completed (on average) about 83 judgments of each kind in each category. Therefore, on average, since there were 500 pages total evaluated, there were are about 18 judgments of each page on each of the two dimensions. (Actually, since only 492 documents were responding to Web queries during the period of the experiment, although all of them were responding when I constructed the list, the number of judgments per document was very slightly higher.) The quality and relevance variables were correlated with one another at 0.747, meaning either one accounted for about 55.8% of the variation in the other. Table 5.1 shows the mean and standard error of the user judgments of quality and relevance of the documents in each of the five categories.

Table 5.1: Descriptive Statistics, Relevance and Quality Judgments, Wisconsin Policy Topics

		relevance			quality		
Policy Area	# subjects	# ratings	mean	sd	# ratings	mean	sd
Economy	23	1699	5.72	2.72	1703	5.41	2.66
Education	22	1809	5.68	2.50	1814	5.31	2.53
Environment	22	1925	5.87	2.36	1931	5.19	2.48
Govt./Politics	20	1777	5.75	2.40	1785	5.00	2.49
Health Care	20	1772	5.90	2.74	1780	5.47	2.65
Total	107	8982	5.79	2.55	9013	5.27	2.56

As one can see from Table 5.1, all of the categories have quite similar distributions, at least as far as the means and standard deviations are concerned. The ratings for relevance are systematically, but only slightly, above those for quality.

Manual consideration of the pages after they were ranked by mean subject led me to the following conclusions about the behavior of the subjects. The subjects tend to rate highly for relevance those pages which are associated with high-status or highly-socially-visible institutions. For instance, the top 6 pages in terms of mean subject relevance ranking in the "government and politics" category are all home pages of state agencies (the Wisconsin Department of Natural Resources, Department of Justice, Office of the Governor, Department of Commerce, State Legislature home page, and the Wisconsin Department of Financial Institutions.).

In the case of the health care pages, the top six pages were pages associated with the University of Wisconsin Hospital (perhaps the most prestigious hospital in the state), Blue Cross/Blue Shield of Wisconsin (perhaps the best-known health insurer), the Medical College of Wisconsin (Wisconsin's other medical school, besides that at the University of Wisconsin, and also very well-known in the state), Community Health Care of Wausau (not that well-known outside Wausau, but a beautifully-designed site), the Wisconsin branch of the March of Dimes (a very well-known organization), and another page associated with the Medical College of Wisconsin. What many of these top pages have in common is that they are associated with very well-known organizations and are very nicely-designed, probably because these institutions have the resources to hire professional graphic artists and Web page designers. On the other hand, those with lower relevance scores tend to be associated with less prestigious or well-known organizations and to not be as well-designed, based on my browsing through the results.

5.1.3 Keyword Models

I built OLS (ordinary least-squares) regression models to attempt to account for the subjects' ratings in terms of the presence or absence of over-represented words on a particular page, and the total number of over-represented word stems (based on their TFIGF score). In an OLS regression, the regression coefficients are calculated by minimizing the sum-squared distance between the regression line and the data points [47]. So, for instance, in a multiple regression of one dependent variable y on three independent variables x_1, x_2, and x_3, we have a regression equation of the form

$$y = \beta_0 + \beta_1 x_1 + \beta_1 x_1 + \beta_1 x_1 + \varepsilon$$

where the β_i are the regression coefficients and ε is the error term. For all the values $y_i$of y, we have a corresponding ε_i, and the $\beta_i$are estimated by minimizing the sum-squared value of the ε_i.

Tables 5.2 and 5.3 show the top 100 over-represented word stems for each of the five categories, in the order of their degree of over-representation relative to a large corpus of random background text. This is equivalent to sorting by descending TFIGF score and then selecting the top 100.

I suspected that the presence or absence of these over-represented word stems on a particular Web page might be related to the subjects' judgments of the page's quality or relevance. However, this turned out to largely not be the case. For each of the five categories, I created two models, for a total of ten models. In each model, the presence or absence of each of the top 25 words in each of the lists in Tables 5.2 and 5.3 was coded as an independent dummy (zero for absent, one for present) variable in an OLS multiple linear regression, and the quality or relevance was the dependent variable. These models had the following form.

5.1.4 Results of Keyword Models and Discussion

When the OLS model is estimated, the statistical software estimates a 95 percent confidence interval for each regression coefficient (beta). This means that the probability is 95 percent that the true beta lies in this confidence interval. A beta is significantly different than zero if the entire confidence interval lies on one or the other side of zero; in those two cases, one can be reasonably confident that the true beta is positive or negative.

Very few of the word stem dummies in these multiple regressions had betas that were significantly different than zero, leading to the suspicion that those that were significantly non-zero occurred by chance, given the large number of variables in the

Table 5.2: Top 100 Over-Represented Word Stems, Wisconsin Policy Areas (Economy, Education, and Environment)

Category	**Top 100 Over-Represented Word Stems**
economy	wisconsin, state, develop, busi, econom, work, research, job, industri, program, economi, studi, year, impact, service, thi, madison, counti, univers, report, product, worker, employ, nation, percent, system, educ, center, school, high, technolog,area, million, resourc, tax, public, commun, milwauke, fund, train, provid, manufactur, depart, compani, health, increas,peopl, wi, support, local, includ, time, student, feder, technic, workforc, cost, labor, inform, make, project, skill, base,incom, growth, region, welfar, govern, institut, benefit, visitor, manag, american, recreat, opportun, famili, creat, care,rate, agricultur, total, plan, relat, annual, gener, bioscienc, home, wage, number, colleg, chang, uw, polisci, futur, top, investrequir, medic, life
education	school, wisconsin, state, educ, program, student, educat, public, colleg, teacher, wi, univers, high, milwauke, district,technic, court, work, home, system, madison, develop, year, children, associat, mpcp, parent, area, amend, privat, learn, nation, site, commu, depart, standard, technolog, includ, skill, uw, resourc, center, provid, counti, make, fax, question,choice, particip, scienc, offic, curriculum, thi, governor, support, requir, institut, people, servic, project, art, teach, middle,board, council, religi, plan, inform, report, instruction, level, establish, train, elementari, librari, activ, claus, busi, law, extension, studi, test, fund, administr, opportun, career, grade, local, time, social, people, purpose, life, effect, web,workforc, benefit, base, thompson, education
environment	wisconsin, water, environment, state, program, resourc, lake, river, manag, site, land, al, environ, develop, natur, sourc,transport, area, qualit, year, pollut, pcb, depart, includ, concentr, fish, protect, project, public, nation, system, feder, contamin, agricultur, work, plan, plant, wi, commun, requir, trade, report, issu, great, point, thi, regul, cryptosporidium,epa, energi, time, polici, madison, local, million, univers, nonpoint, research, control, inform, conver, gener, provid, impact,support, studi, industri, speci, wast, law, act, mine, group, anim, effect, fund, educ, increas, counti, altern, watersh, activ,base, reduc, peopl, wildlif, home, level, product, carp

Table 5.3: Top 100 Over-Represented Word Stems, Wisconsin Policy Areas (Government/Politics and Health Care)

Category	**Top 100 Over-Represented Word Stems**
government and politics	wisconsin, state, counti, govern, polit, public, citi, tax, parti, site, madison, republican, local, campaign, year, depart, servic,inform, democrat, work, web, time, governor, district, thompson, vote, elect, peopl, home, wi, senat, milwauke, feder, court, legisl, reform, nation, candid, informat, program, make, monei, committe, thi, mail, law, search, resourc, green, univers, congres, report, issu, school, member, lafollett, town, student, board, contact, gener, legislatur, system, commun, budget,meet, interest, tommi, includ, support, dai, narrat, relat, busi, group, office, dave, progress, hous, list, bill, repres, commiss,health, tim, citizen, plan, polisci, famili, offic, colleg, record, call, find, presid, financ, provid, industri, offici, educat
health care	health, wisconsin, care, state, tobacco, program, servic, product, provid, public, inform, hospit, medicaid, nicotin, research,medic, percent, smoke, compani, commun, data, children, insur, home, year, center, patient, includ, counti, famili, group, wi, system, madison, market, long, cigarett, nation, clinic, plan, physician, defend, report, depart, polisci, informat, thi, base,manag, term, rate, develop, gener, md, time, increas, studi, network, fund, effect, site, industri, school, associat, support,continu, cost, nurs, milwauke, diseas, institut, requir, level, length, area, result, feder, work, peopl, univers, resourc, relat,administr, number, grant, make, issu, project, cancer, person, popul, order, access, addict, contact, benefit, ag, part, agent, primari

regressions.

However, for the record, here is a list of those word stems in each category that did have dummy betas significantly different from zero: the word stem “Wisconsin” in the model for relevance in category 1; the same word stem, for quality, in category 1; the word stem “wi” (the postal abbreviation for Wisconsin) in the model for relevance in category 2, and again for quality in category 2; the word stem “home” for quality in category 2 (but not for relevance); no word stems for either quality or relevance for category 3; the word stem “site” for both quality and relevance in category 4; the word stem “parti” for relevance in category 4 (but not for quality); and the word stem “research” for category 4 for relevance (but not for quality). This represents 9 word stems with significant betas out of a total of $25 * 5 = 125$ keywords tested tested in 2 models each, for a possible 250 relationships. 9 out of 250 (3.6 percent) is not a very impressive number, and is probably around the number one would expect by chance, especially since there appears to be nothing special about the keywords that do have non-zero correlations. So I have not demonstrated any ability of overrepresented keywords to predict quality or relevance evaluations by subjects. It is likely that subjects are doing something much more sophisticated in their evaluations than what such models capture.

5.1.5 Results of Models Based on Length and Content

I also created six univariate regression models, attempting to predict the quality or relevance of a page based on the page’s raw length in bytes (as downloaded from the Web) (two models here), the number of unique word stems on the page (two more models here), and the number of the top 100 overrepresented word stems that the page contained (two more models here), on the theory that more complex pages might earn higher relevance or quality scores. Here, each of these models had either the quality or relevance as the dependent and one of the other three variables as an independent variable. The formulation of a OLS univariate regression model is similar to that of a OLS multivariate regression as described above, however, there is only one independent variable. However, the system still minimizes the sum-squared differences between the predicted (model) and actual (sample) values. This is an example of a simple OLS regression equation of one dependent variable y on one dependant variable x:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

The residual ε represents the difference between the predicted value and the actual value.

However, none of the six models I created had betas that were significantly different from zero.

5.1.6 In-link Model Contruction, Results, and Discussion

I also created two OLS regression models in which the total number of in-links (from all 492 pages in the set itself) to a given page was used to predict the mean subject quality or relevance of each page. Here, the models for the five categories could be combined into one, because all of the variables are uniform across the five sets. I found a very weak effect here. For the relevance model, the beta was 0.07, but this was significantly different from zero (in terms of a 95 percent confidence interval; that is, the entire 95 percent confidence interval was above zero); however, the adjusted r-squared of the model was 0.015. The adjusted r-squared is a measure of the share of the variance that is accounted for by the model. For the quality model, the beta was 0.06, but again it was significantly different from zero (again, in terms of a 95 percent confidence interval); the adjusted r-squared value was 0.010.

Another proxy for the number of in-links into a page is the total number of in-links reported by the various search engines. A good proportion of the search engines have a query format that allows you to get the number of pages pointing into a given page and a list of the pages. For the 492 pages in the Wisconsin policy group, I found the number of in-links given by Altavista and by Google. Google reported that 206 of the 492 pages had a positive (that is, greater than zero) number of in-links; Altavista reported 231 of the 492 pages as having a positive number. Note that in both cases, this represents a minority of the pages as having in-links. This could be due to two factors; that these pages do not really have any pages pointing to them, or that the search engines are doing an imperfect job of spidering (which is almost certainly the case). It is actually hard to tell which, because a search engine needs a starting point to spider, and if there is a clique of pages that is disconnected from the rest of the Web, it may take some time for this clique to have a pointer off of a page that the engine already knows about.

In the case of both Google and Altavista, an OLS regression model based on an independent variable which was the log of the number of in-links performed better than a linear model based on the number of in-links. This is likely due to the fact that in-link distributions are likely to (approximately) follow Zipf's law, which predicts a hyperbolic distribution; taking the log of the number of in-links flattens this distribution, which will work better with a linear regression. The model based on a linear dependent variable is as follows:

$$Relevance(page) = \beta_1 Inlinks(page) + \beta_0 + \varepsilon$$

The model based on a log dependent variable is as follows:

$$Relevance(page) = \beta_1 log(Inlinks(page)) + \beta_0 + \varepsilon$$

$Inlinks(page)$ is as reported by either Google or Altavista; separate linear and log models are estimated for each search engine, for a total of four models.

In the case of the Altavista log model predicting mean subject judgments of page relevance, the beta, estimated at 0.23, was significantly different from zero, and the adjusted r-squared value was 0.11. The respective numbers for the Altavista linear model were 0.00016 and 0.017. For the Google log model, the values were 0.24 and 0.13. For the Google linear model, the numbers were 0.00040 and 0.058. Each of the 95 percent confidence intervals for each of the betas in these models was fully above zero. So, the Google log model did best in accounting for variance in subject judgments of page relevance, although only slightly better than the Altavista log model, and probably not significantly so. For the log models, the observations with zero in-links were dropped, since they have undefined logs, so I also dropped these observations in the linear models to make them comparable.

For the prediction of page quality (rather than relevance; otherwise, the models take the same form as above), the numbers were as follows: for the Altavista log model, a beta of 0.30 and an adjusted r-squared of 0.15; for the Altavista linear

Table 5.4: Linear and Log Models Relating Page In-Links to Subject's Mean Judgments of Quality and Relevance

Search Engine	Dep. Variable: Linear/Log Page In-links	Indep. Variable: Quality/Relevance	Beta	Adj. R-squared
Altavista	Log	Quality	0.23	0.11
Altavista	Linear	Quality	0.00016	0.017
Google	Log	Quality	0.24	0.13
Google	Linear	Quality	0.00040	0.058
Altavista	Log	Relevance	0.30	0.15
Altavista	Linear	Relevance	0.00020	0.024
Google	Log	Relevance	0.31	0.12
Google	Linear	Relevance	0.0052	0.061

model, 0.00020 and 0.024; for the Google log model, 0.31 and 0.12; and for the the Google linear model, 0.0052 and 0.061. Again, each of the 95 percent confidence intervals for each of the betas in these models was fully above zero. So, again, in this case, the Google log model performed slightly better than the Altavista log model. But all these numbers are quite similar to the numbers that were found for page relevance, which is not surprising given the high level of correlation between quality and relevance. All these regression results for quality and relevance are summarized in Table 5.4.

The adjusted r-squared values for the four log models mentioned above range between 0.11 and 0.15. Although the relationships are significant based on the confidence intervals of the betas, this is not a particularly impressive accounting for the variance in the relevance or quality scores. Thus a ranking based on pure in-links does not do a good job in reflecting subjects' judgments of placements into a particular category. This is not surprising, because these all in-links from all other pages on the Web that the search engine knows about, not just those specific to the category.

Google's strategy is not likely to be much better with respect to a particular query, even though it is more sophisticated than the raw number of in-links, but measures something akin to the total network inflow to a page. This is because it must combine this PageRank value with a standard keyword-based ranking, unrefined keyword queries are not likely to produce high precision or recall, and the PageRank value is global to all pages on the Web, so it is not relative to the query. But, of course, pages need to be ranked differently depending on what the query is.

There is a strong correlation between the number of in-links reported by Altavista and those reported by Google to these 492 pages. The correlation is 0.868 (between the linear measures) if the zero in-links are kept, and 0.870 if they are dropped. The correlation between the log measures (with the undefined (log zero) values dropped) was 0.737.

The mean number of in-links reported by Altavista to the 492 pages in this set was 134.6, but this mean is high because of the highly skewed distribution of in-links per page. For instance, the median number of in-links is zero, because over half the pages have no in-links that Altavista knows about. The number of in-links at the 75th percentile is only 25. However, the page with the largest number of in-links has 12,967! If you drop the pages with zero in-links, the mean rises to 286.7 (and the number of observations drops from 492 to 231).

For Google, the mean number of in-links was 95.2. As in the case of Altavista, the median was zero. The number of in-links at the 75th percentile was 42.5, and the maximum number of in-links was 3950. (The page with the maximum number of

Table 5.5: In-Link Statistics for the Wisconsin Policy Pages, as Reported by Altavista and Google

	Altavista	Google
Number of Pages with One or More In-Links (of 492)	231	206
Mean In-Links	134.6	95.2
Mean In-Links excluding Pages without In-Links	286.7	227.3
In-Links at 75th Percentile	25	42.5
Maximum In-Links	12,967	3950

in-links was not the same in each case). If you drop the pages with zero in-links in this case, the mean rises to 227.3 (and the number of observations falls from 492 to 206).

These data about the two search engines are summarized in Table 5.5.

So, given the fact that Google has both more pages for which it does not know any in-link, and a lower mean for the remaining pages, it appears that Altavista has better coverage of at least this (Wisconsin policy-related) section of the Web. This is gives us a hint of how one might compare the coverage of search engines, since they certainly cannot be taken at their word (and certain factors, such as the high prevalence of dead links and dynamically-generated pages, make a comparison of counts difficult, since each search engine is likely to use a different counting method). Running a similar experiment with randomly selected pages (rather than the manually-selected ones used in this case) might give a better method for comparing the coverage of search engines. However, it is difficult to find a truly random selection of pages, because all such sets have to be drawn from an existing database (e.g. search engine or directory), which is likely to be biased in its representation of the Web as a whole.

It is also likely that there is a correlation, perhaps a strong one, between the relevance and quality scores of pages and those of the pages that they link to; it would not be difficult to collect such data, but I have not yet done so. Such correlations would be useful in determining where to place new pages spidered off of an existing set of pages that have already been placed in a category, by using the quality or relevance of the linking pages to predict the quality or relevance of the new page.

5.2 Review of the Contributions of this Chapter

This chapter differs from the rest of the chapters in this thesis in that it makes use of data drawn from sample subjects. Direct measurement of subjects' judgments of quality and relevance on a set of pages on particular topics represent a good proxy for how subjects would rank those pages. Ideally, if one could predict the mean value of the quality and/or relevance to a topic of a particular page, one would know where to rank it relative to others.

The first two contributions in this chapter are negative ones. I was not able to predict quality or relevance of pages using a vector of dummies representing the presence of each of the 25 most salient keywords in a group, ranked by TFIGF score. Neither was I able to do so using the size of the pages, the number of the top 100 most salient keywords, or the number of unique keywords.

The final contibution was more positive, although the effect was weak. I showed that the log of the number of in-links was a significant predictor of the mean quality or relevance scores, although the models accounted for only between approximately 11 and 15 percent of the variance in the dependant variable. The effect may possibly be amplified by trying other models, but the more models one tries, the more one is guilty of data mining.

I suspect that Google PageRank scores would not do much better, although there is no way to test this, because Google does not release its numerical PageRanks to the public. Even if the Google PageRank scores did twice as well—which seems unlikely—they would be accounting for at most 30 percent of the variance. I suspect much higher-level processes are involved in the assessment of Web pages by human subjects, such as the design of the page, the prestige of the person or institution producing it, and more complex linguistic processing of its content.

The main contribution—and point—of this chapter is that there is a lot of room for improvement in page ranking schemes, since in-link counts constitute a major component of the way that it is done now, and they do not correllate well with subjects' rankings. Combining the empirical methodology of this chapter with experiments in ranking techniques such as those employed by Active Portal (and improvements thereupon) found in Chapter 4 could be a potentially fruitful line of research in developing optimal page ranking techniques. This, however, has to be combined with techniques for semantic separation through classification or clustering, as discussed in Chapters 3 and 6 respectively.

Chapter 6

Clustering Web Pages

In this chapter I investigate the clustering of Web Pages. Clustering is an unsupervised technique of associating related items, while classification is a supervised technique. While human supervision, which often involves providing labelled examples of classes, often provides better results, there is much that can be accomplished without supervision. The Web provides much implicit human intelligence in terms of the link structures between pages; two pages that are linked are much more likely to be semantically linked than two randomly-selected unlinked pages. This link information, as well as the text on the pages themselves, can be usefully mined to impose further organization on the Web.

In this chapter, I examine several approaches to Web page clustering. First, in Section 6.1, I examine the idea of using, as clusters, connected components of the Web viewed as an undirected graph, after pruning out highly-globally-referenced pages that are unlikely to be on specific, focussed topics. Second, in Section 6.2, I look at the extent to which WordNet can be used as a tool for clustering Web pages. Third, in

Section 6.3 I apply HAL-like [17] semantic networks constructed out of the text of search engine results to the task of clustering these results.

6.1 Connected Components and Web Navigation

In this section, I examine the degree to which connected components of pages on the Web can be useful as clusters. Initial experiments found that semantically-unrelated pages tend to be connected through highly-globally-referenced pages, so I computed a list of these highly-globally-referenced pages and removed them from the sets of pages that were being clustered before computing the clusters. Since I had this list of these highly-referenced pages, though, I thought it would be interesting to see if they follow a Zipf's law distribution.

6.1.1 Finding Highly-Referenced Pages and Examining a Zipf's Law Model

Individual Web sites tend to be organized, when drawn as graphs, like trees or cliques, or some combination thereof. Even if they are organized as trees, there are often back links to pages higher up in the tree, or to the home page. Due to the power law distribution of Web site size [53], a large number of the links on the Web are links into a relatively few Web sites, such as microsoft.com, netscape.com, aol.com, yahoo.com, etc. As we will see later in this chapter, identifying these most highly-referenced sites is helpful in Web clustering, because semantically-unrelated cliques (connected components) of pages tend to be connected together via globally very-highly referenced pages.

As we have seen (for instance, in Section 1.1), a Zipf's (power) law distribution is a distribution where the nth highest-ranking item has a frequency directly proportional to $1/(n^k)$, where k is an exponent close to one. Other research has examined the degree to which Web requests from a given user community are distributed according to Zipf's law, and the results have been mixed; for a review of this, see [13]. In addition, based on a large Web crawl, Albert, Jeong, and Barabasi found that the probabilities $P_{out}(k)$ and $P_{in}(k)$ that a given document on the Web has k out-links and in-links both follow power laws, with exponents k of 2.45 and 2.1 respectively [3].

Here, I examine a related issue: when a set of Web pages are ranked in terms of their in-link count, do the ranks follow Zipf's law? I do this because I generated such a sample of pages for a use in a later section of this chapter, and I thought it would be interesting to see if the pages followed Zipf's law, since they were on hand in any case.

Hypothesis

The most highly referenced pages follow a Zipf's Law distribution with respect to the number of in-links each of them has.

Method

In order to determine some of the most commonly referenced sites on the Web, I used Yahoo!'s random page selector, which (presumably pseudo)-randomly selects a page out of their hierarchy. At this writing, this selector was found at http://random.yahoo.com/bin/ryl. While this system does not (in all cases) return the URL of the page that it returns, one can than follow the unique absolute links off that site. I wrote a program that repeatedly called this selector and then dumped all of the absolute links it found in each returned file into a big file, until I had accumulated 100,000 links (this was an arbitrary number; I figured this number of links would reveal at least most of the most referenced ones, if not in the correct order). I then counted how many times each link was referenced in the list of 100,000.

I attempted, using OLS linear regression[1], to fit this list to a Zipf's Law model; that is, a model in which the frequency (in terms of in-links) of a particular item (here, a Web page URL) is directly proportional to the inverse of its rank in the list. Consider Zipf's Law:

$$F = c\frac{1}{n^k}$$

where c and k are constants, F is the inlink-frequency of each page in the list of pages ranked by in-link frequency, and n is the rank of each item on the list. Taking the log of each side, we get:

$$log(F) = -klog(n) + log(c)$$

which has a form similar to a regression equation

$$y_i = b_1 x_i + b_0$$

where $y_i = log(F)$, $b_1 = -k$, $x_i = log(n)$, and $b_0 = log(c)$. Thus I estimate the OLS linear regression with $log(F)$ and $log(n)$ as the dependent and independent variables, respectively.

[1]OLS linear regression is briefly described in Section 5.1.3.

Results

I found that such a model fit the data (the first 1000 items in the list, that is) with a beta of -0.61 ($k = 0.61$) and an adjusted r-squared value of 0.97. However, the value of k is quite different from unity (Zipf's Law is sometimes defined as having a value close to unity), although it is on the same order of magnitude. This is an excellent fit. In Figure 6.1, the data are plotted; as you can see, the relationship is quite linear, except for the downward slope of the tail at the extreme right. This shape is similar to that shown in Albert et al. [3].

Many of the pages that had the highest frequencies in this list were not what I would have expected, but that is probably due to the particularities of Yahoo!. For instance, some of the top frequencies were to pages on the New York Times Web site, but this is probably due to the partnership that Yahoo! has with that site. However, many of the pages that you would expect to be most frequently referenced are near the top of the list of 61,130 unique URLs found among the 100,000 links gathered from the pages. For instance, www.microsoft.com is #78, www.apple.com is #320, www.cnn.com is #334, and www.ibm.com is #1872.

I also did an extraction of 100,000 URLs using random links drawn from a list of about 1.23 million Dmoz links that I had downloaded from www.dmoz.org. Following about 13000 of these links (some of which were dead) yielded the 100,000 links. This list looked quite a bit different than the one drawn from Yahoo!; as in the case of the Yahoo! list, the top links are not Amazon, Yahoo!, IBM, etc. as one might expect, but considering the same pages as in the Yahoo! list, www.microsoft.com is #50, www.apple.com is #361, www.cnn.com is #482, and www.ibm.com is #464. So this (admittedly small) sample is approximately in the same order, although the final two

Figure 6.1: Log(Rank) Versus Log(In-Link) Frequency for the Top 1000 Pages in Terms of In-Links within a Sample of 100,000 Pages Randomly Drawn from Yahoo!

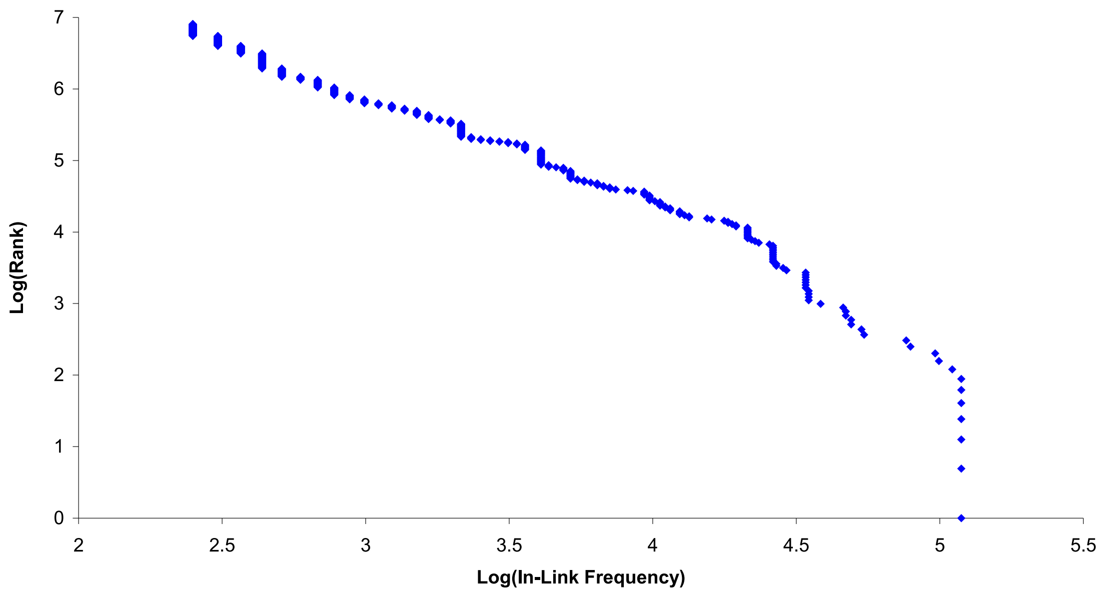
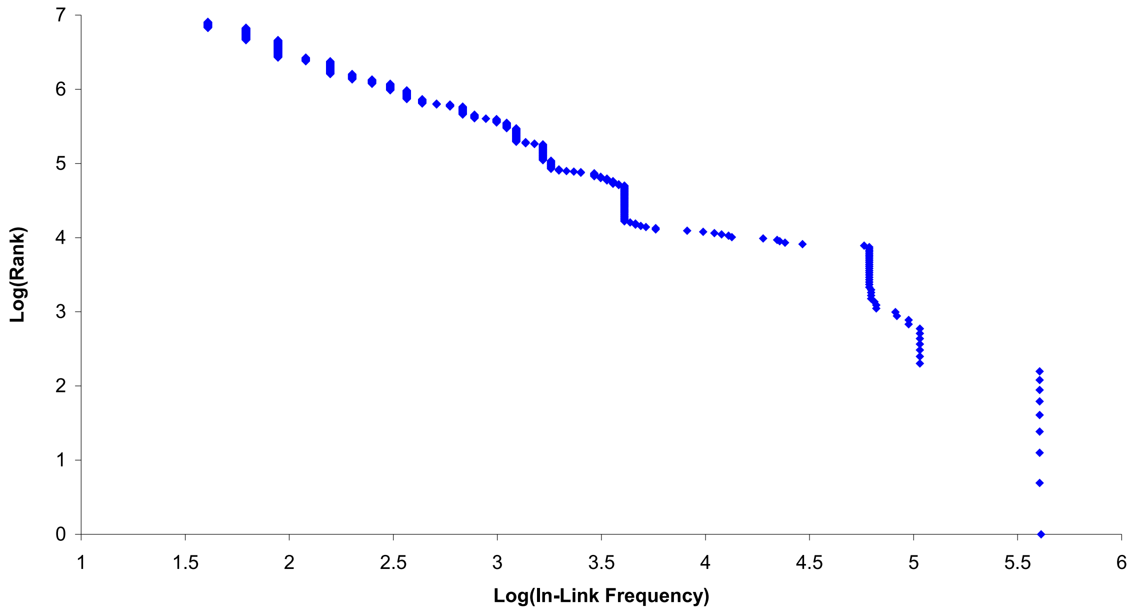


Figure 6.2: Log(Rank) Versus Log(In-Link) Frequency for the Top 1000 Pages in Terms of In-Links within a Sample of 100,000 Pages Randomly Drawn from Dmoz

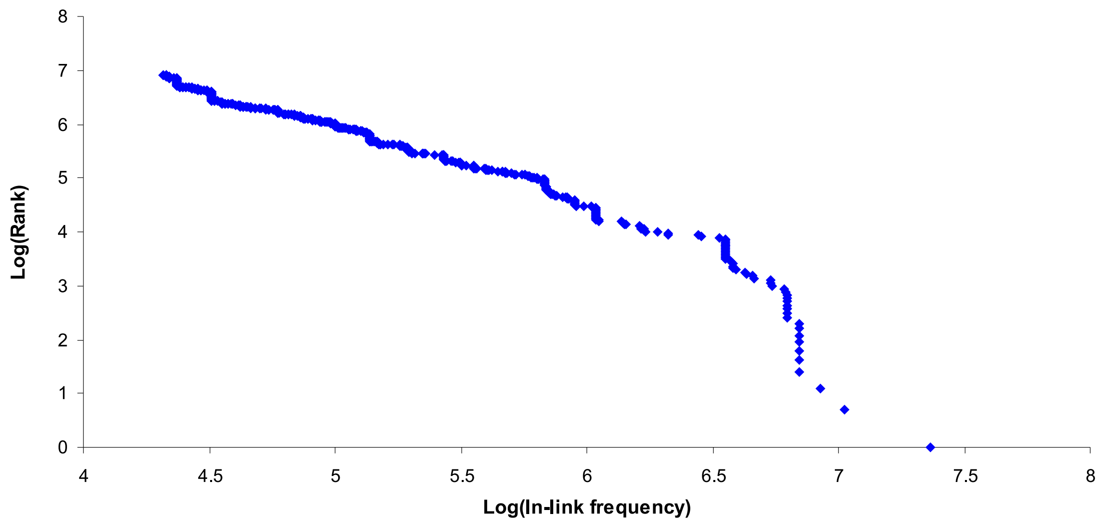


are interchanged. A better model would be constructed by combining both and using more pages from each.

Fitting the same Zipf's-law-based log-log regression model to the top 1000 frequencies in the list, we get a beta of -0.83 ($k = 0.83$) and an adjusted r-squared value of 0.97. This is again an excellent fit; the data are plotted in Figure 6.2. Again, the value of k is quite different from unity, although not so much as above.

I also tried to use a larger sample of one million links, which 500,000 were drawn from links off of 14,152 random Yahoo! pages and 500,000 were drawn off of links from 13,014 random Dmoz pages. A log-log fit of Zipf's law to this sample gave a beta value of -0.63 and an adjusted r-squared value of 0.95, again an excellent fit.

Figure 6.3: Log(Rank) Versus Log(In-Link) Frequency for the Top 1000 Pages in Terms of In-Links within a Sample of One Million Pages, Half Randomly Drawn from Dmoz and Half from Yahoo!



The data are plotted in Figure 6.3. The value of k once again differs from one.

Since this is the largest sample I have, I used it to determine the highest-frequency pages on the Web for the purposes of disconnecting cliques of pages connected by highly referenced pages, for the purposes of disambiguation, as described in more detail below.

Discussion

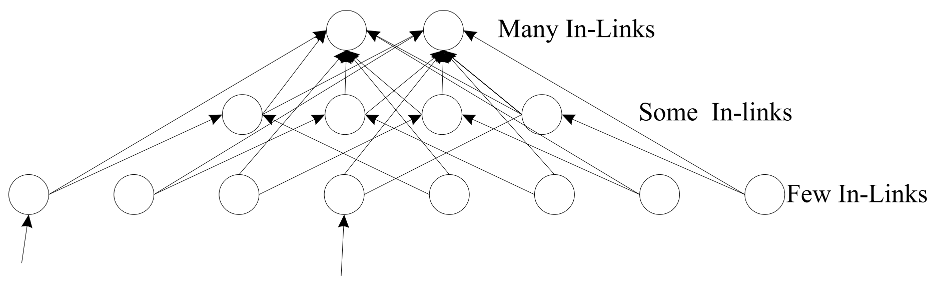
These experiments have validated Zipf's Law for the particular sets of pages I have been using. This is interesting because the formulation of the Zipf's Law relation (in terms of relation between the number of in-links to a page and the page's rank with respect to that number) is different from that seen earlier in the literature, and therefore the Zipf's Law exponents are different from those obtained, for instance, by Albert, Jeong, and Barabasi [3].

Another way to attempt to extract a random sample of pages from the Web is to pass random English words to search engines as queries and collect some fixed number of URLs per query. If you pass the search engine 100,000 words and collect 100 pages per word, that will give you a sample of 10,000,000 URLs, and you can then find an even larger set of pages to which these point. I have not tried this approach yet, though. The ideal approach, would be to elicit the support of some large search engine such as Google or AltaVista to give out a large pseudo-random sample of its page list, and then frequency counts of the pages these point to can be computed. But the large search engine companies may not be willing to divulge this information to outside researchers, for proprietary reasons. Another good source of information for separating out connected components on topics would be to exclude those pages from the graph that are among, say, the top 1,000 pages in terms of their global Google PageRank. However, Google does not divulge these pages. Perhaps if it did, other search engines would favor these pages as well, which Google does not want to happen.

6.1.2 Hierarchies of Pages and Connected Components

The Web can be viewed as organized in a hierarchy. At the bottom are individual home pages, pages of small companies and institutions, etc. Higher up are topical "authorities" and "hubs." In the terminology of the Clever project [58], these are referenced many times ("authorities") or contain large numbers of references

Figure 6.4: Part of the Web Visualized as a Hierarchy



("hubs"). At the top of the hierarchy are the most globally-referenced pages, such as www.yahoo.com, www.amazon.com, www.nytimes.com, www.cnn.com, etc. An idealized view of such a hierarchy is shown in Figure 6.4. Of course, this hierarchy is only an idealization; in reality there are many downward and lateral links. However, hierarchies give some insight to the structure of the Web. However, as we have seen above, Albert, Jeong, and Barabasi found that the probabilities $P_{out}(k)$ and $P_{in}(k)$ that a given document on the Web has k out-links and in-links both follow power laws, with exponents k of 2.45 and 2.1 respectively [3]. This indicates that exponentially-diminishing shares of the pages on the Web have large numbers of in-links and out-links. This is consistent with the view of the Web as a hierarchy.

Since the Web is divided into many topical interest groups, there are pages that are authorities or hubs on almost any conceivable topic. The hierarchical nature of the Web is not really a computing phenomenon; it is a social phenomenon, having to do with the fact that there are many hierarchies of status, power, and financial capital in society; the Web reflects all of these, like all media, see, for instance, [77]. It is also something of a sociological phenomenon, that comes from the fact that if one person sees a site that she likes, and links to it, this increases the probability that someone else will see that site and link to it as well.[2]

For instance, there are many pages on the Web which are maintained by Jaguar (automobile) dealers and enthusiasts. Many of these pages point to the home page of the Jaguar automobile company, which is therefore an authority. Others contain many pointers to other pages of dealers and enthusiasts, and therefore serve as hubs. Note that the Jaguar company site is also a hub, since it links to many dealer Web sites. Authorities (and to a lesser extent hubs) are likely to be listed in directory sites, so this places the directory site higher up in the hierarchy.

Also, many of the pages lower in the hierarchy are likely to point to popular search engine and directory sites, as well as other popular sites such as Amazon.com. The latter site encourages linking by paying a percentage of sales to the linking sites when the sales were made as a result of the link. Thus, much of the inflow to sites comes from pages lower down in the hierarchy. In addition, there are sites that host large numbers of pages, such as the Geocities community site (now part of Yahoo!), and these sites get major inflow by virtue of the number of pages that they host. Of course, the Web is not organized as anything close to a complete hierarchy; not only are there pages that do not participate in a hierarchy, there are many lateral links between sites at the same level of the hierarchy, among pages that do participate in a hierarchy to some extent. Thus, it embeds both a hierarchy and other, lateral, links.

One sees this in groups of pages run by groups of enthusiasts. However, looking at the subset of links that creates a hierarchy allows one to see how topically unrelated portions of the Web are connected to one another. They are typically connected via

[2]This process could potentially be modelled, if one assumes something about the distribution of Webmasters' quality assessments and the probability that those quality accessments would be translated into new links by the Webmasters. Such a model would have to take into account the differential overall proclivity of Webmasters to create new links.

a large site which has no particular topic, but is used by the entire Web (such as Yahoo!.) The Web can be thought of as a directed graph, and directed graphs can be separated into strongly connected components.

Each strongly connected component has the property that any two nodes A and B in it are mutually accessible; i.e. you can get from A to B via directed links, and back again via other directed links. However, it is not possible to get from any node in one component to any node in another component and back again. An undirected graph can be separated into connected components, which are simply subsets of the nodes of the graph, and which have the property that one can get to any node from any other in each subset by a path of one or more edges.

The Web can also be thought of as an undirected graph, if one declares, for this purpose, that two nodes are connected (by an undirected connection) if there is at least one (directed) connection between them in the directed graph representation of the Web. Connected or strongly connected components can be used to separate out regions of the Web that are likely to be semantically related. A method of computing connected components is given in Figure 6.5. In order to make them work better in a Web application, it is best to remove links to portal sites or other highly referenced sites, such as Yahoo! or Excite, because these sites tend to be quite strongly connected in a bi-directional manner and therefore can connect semantically unrelated material. In order to do this, one simply removes all links involving these highly-referenced pages, and then recomputes the components as given in Figure 6.5.

Also, within Web sites, it is best to add a link back to the home page from other pages, if it is not present, for the purpose of computing strongly connected components. This prevents nodes within Web sites that are directed trees from being

Figure 6.5: A Method of Computing the Connected Components of a Set of Web Pages W

1. For each undirected link (p_1, p_2) between two Web pages $p_1$and p_2 in W:
   (a) If neither p_1 nor p_2 have been placed in a component, create a new component containing p_1 and p_2
   (b) If p_1 is in a component, but p_2 is not, add p_2 to the same component in which p_1 is found.
   (c) If p_2 is in a component, but p_1 is not, add p_1 to the same component in which p_2 is found.
   (d) If p_1 and p_2 are in different components, merge these two components into a single component.
2. Return the set of components so generated.

considered singleton strongly connected components. Fortunately, the URL structure of many Web sites is organized as a tree, so it is possible to find such tree-like structures. A link back to the home page from internal nodes creates cycles making it possible to follow links from any page to any other, and makes the entire site a strongly connected component.

6.1.3 Computing Connected Components: An Experiment with a "Jaguar" Query

For simplicity, I have experimented with connected components, viewing the a subset of the Web as an undirected graph. As a preliminary experiment (more detailed, comparative experiments are described below), following Kleinberg and the Clever Project with which he is associated [58], I have used the query "jaguar" when passed to the search engine Altavista. Again following Kleinberg, I then augmented the results of the query (the primary result) with secondary pages which either point to or are pointed to by pages in the primary result. These additional pages may be particularly relevant to the query because they are likely to contain hub or authority pages, in the terminology of the Clever project. Also, they are likely to make the graph composed of the primary and secondary pages better connected than the primary pages alone would be. In the case of the "jaguar" query, these result pages appeared to be on the following subjects: the jaguar (the wild animal), the automobile, the Jacksonville Jaguars football team, a chemistry software package named Jaguar, and the Atari Jaguar video game.

I then computed, using exhaustive search of all of the search results, all the connected components within the undirected graph representing all these primary and secondary pages. The initial such computation contained one large component that contained references to pages about the chemistry software, the animal, the automobile, and the video game. This led me to wonder which pages were connected to such semantically unrelated regions of the Web. So, I used a breadth-first search to find the path from one node to another. I found that these semantically unrelated sites were connected through nodes, such as the Internet Explorer home page and the "Dilbert" home page (about a cartoon which is very popular on the Internet), which are semantically unrelated to any of them but are linked to by many pages. The only one of the linkages that was marginally semantically linked to both sites that it connected was the Nascar home page, which has to do with auto racing. It was linked to one of the Jaguar automobile pages as well as a page where a game hobbyist had implemented a video game version of an auto race.[3]

A Strategy: Eliminate Highly-Referenced Pages

However, the Nascar page, the Dilbert page, and the other pages that linked the semantically-unrelated components, are all among the most-frequently referenced pages on the Web. This suggests the following heuristic for successfully separating semantically-related components of the Web. Use a large sample of random Web pages to compile statistics on the Web pages to which other pages most frequently link. Impose a cutoff; say the top 500 sites that receive links.[4] Then remove these sites from the graph from the point of view of computing components. The best place to make the cutoff can be determined experimentally. This modified algorithm is described in Figure 6.6.

In my experiment with the "jaguar" query, removing these frequently-referenced pages broke the graph into components in which all the pages in most of the particular components were about one particular sense of "jaguar." The details of the results of this process are discussed in Section 6.1.4. There were also pages within a given component that were irrelevant to the query, but happened to be linked off of a page that was. After the connected components of the graph generated by response to a query are computed, one is still confronted with three problems.

The first problem is that some of the components contain pages that are com-

[3]Prof. Jude Shavlik of the University of Wisconsin has pointed out to me that a dictionary page for say, "jaguar," that contains links to pages of that are about each meaning, would connect the connected components. Such dictionaries would therefore have to be explicitly excluded via hand-coding, or a method for detecting dictionary pages would need to be devised.

[4]This is an imperfect method to find the Web pages that are most frequently referenced on the Web. Ideally, we would visit all the sites on the Web, but this pseudo-random sample is the best we can do.

Figure 6.6: Computing Connected Components of a Set of Web Pages W, Omitting the Globally Most Highly-Referenced Pages

1. Obtain a large random sample of Web pages (via a service such as Yahoo!'s random page generator). Call this set X.
2. Consider the links off of X. Call the set of such links Y. Count, for each link in Y, the number of pages in X pointing to it. Sort Y by this in-link count, in descending order. Consider the top n pages in Y, where n is a tuning parameter. Call this set of top pages Z.
3. Remove any pages in Z from W, and then follow the procedure for computing connected components given in Figure 6.5.

pletely or partially irrelevant to the query. The second problem is that some of the components still contain pages from more than one disjoint meaning of the query (e.g., in the case of the components of the "jaguar" query, still contain pages referring to the animal and the car). The third problem is that some of the components are separated, but refer to the same meaning of the query—there just happens not to be any connection. Since the link structure of the Web is constructed by large numbers of people, over whose activity one has no control, these problems are likely to recur. However, one can do semantic and textual analysis in order to alleviate them. A simple way to deal with these problems involves the term-vector approach common in the information retrieval field. One can treat all the documents in each component as a single document and compute the term vector for the combined document, dropping very frequent terms. Or the centroid (average) of the term vectors can be used, after normalizing each term vector, so as to treat all documents equally, both short and long.

As we have noted earlier, clustering can be computationally expensive; many algorithms involve $n \log(n)$ term-vector dot product steps, where n is typically the number of documents in question, which can be expensive to do on the fly when you want to return search engine results to a searcher in a clustered manner. Some newer algorithms have a near linear time complexity, but still typically require multiple iterations through the documents [56]. Using connected components and treating them as single documents for the purposes of clustering can reduce the effective value of n significantly, since the Web is already telling you, through its connections, which pages are semantically related to which other pages.

Components can be combined if their combined term vector is similar. So, for instance, if a particular component is about the animal sense of jaguar, it will tend to contain words such as "zoo" and "cat." Other components containing these words can be combined with that component.

In addition, pages within a particular component can be dropped if their term vector is too dissimilar with that of the component as a whole. This can be a way of getting rid of pages that are semantically unrelated to the query in any of its senses. If a particular page is not similar to the average of the component that it is currently in, but is more similar to another component, it can be moved to that component, rather than discarded. Or, a new component can be created.

Another way to deal with these problems involves the use of WordNet (or a similar semantic network). Individual documents or sets of documents can be characterized by the set of "semantic neighborhoods" within WordNet that their terms imply. I define a semantic neighborhood in WordNet as a group of words connected by the various word relations (hyponymy, hypernymy, synonymy, and meronymy) that WordNet supports. A document is associated with a particular semantic neighborhood if it has several words that fall within that neighborhood. Typically, a document will be associated with several semantic neighborhoods within WordNet. Note that the use of WordNet overcomes a typical drawback of the use of term vectors; the fact that different terms can refer to the same concept. In WordNet, for instance, both the word "car" and the word "automobile" will bring up the same semantic neighborhood, because these terms are both members of the same synset (set of synonyms).

Connected components can be combined if they have one or more semantic neighborhoods in common (again, this is something that needs to be experimentally tuned). Documents can be dropped from a connected component if their semantic neighborhoods are too different from those of the component as a whole. Even though WordNet does not contain some of the proper names that are involved in Web queries, it can still be used to classify the results of those queries. Thus, even though WordNet does not know that a Jaguar is a kind of a car, it still can distinguish pages that are about the animal from those that are about the car, since these evoke different semantic neighborhoods in WordNet.

6.1.4 Semantic Disambiguation of Three Ambiguous Sets of Query Results: A Detailed Look

Let us consider three ambiguous queries in more detail. (Of course, most queries are ambiguous, but we are simply considering three simple ones.) The ones I have chosen are "Jaguar" (as above), "Lincoln," and "Ford."[5]

[5]There are actually some semantic connections between these words; for instance, the Jaguar automobile company is now owned by the Ford Motor Company, and Lincoln is a brand of Ford automobile. And Abraham Lincoln was shot at Ford's Theatre (not the same Ford).

Hypothesis

Separation into connected components while removing links to highly-referenced pages can be an effective method to semantically separate groups of pages.

Data Set

For a "jaguar" query, Google returns 1,060,000 results,[6] which means that this many pages on the Web contain that word (of those that are in Google's database). The first 9 results are about the car. The 10th result is about a "Star Trek" ship named the "USS Jaguar." Further down the list, there are pages about the Atari Jaguar (the video game), the Jaguar chemistry software, some software called Jaguar developed at Berkeley which extends the Java programming language, the animal Jaguar, a resort in Belize named after the animal, a site featuring scantily-clad women posing with Jaguar cars, and a page about the German movie "Aimee and Jaguar," a fan page for the TV show "The X-Files" put together by someone calling himself Jaguar, a pornographic site called "Jaguar's Bikini," a site called "Jaguar Sun" about Mayan culture, and the home page for something called the "Jaguar Rubber Stamp Company." This is what I found by looking through the first 120 results; I am certain that I would have found more types of pages if I had continued looking.[7]

For a "Lincoln" query, Google returns 3,590,000 results. (Note that this indicates that "Lincoln" is a more common word on the Web than "jaguar," and therefore probably more common in English texts in general, because the total corpus of text

[6] This is what Google says; it is obviously rounding to the nearest 10,000.

[7] The richness of the use of proper names and metaphorical uses in natural language, as illustrated by this one case, shows how daunting a task the enumeration of the possible meanings of a word can be. Such an enumeration has been undertaken by the WordNet group.

on the Web is probably comparable to a general printed English corpus, although it may have a higher frequency about words about certain topics, such as technology.)

The first result is about the Lincoln automobile. The second result is about Abraham Lincoln. The third result is about Lincoln, Nebraska. Further down the list are pages for Lincoln University (in Pennsylvania) and Lincoln Center (the performing arts center in New York City). What is interesting about this, semantically, is that all of these entities are named for Abraham Lincoln, but in the mind of the searcher, that probably makes little difference. That is, if the searcher is looking for pages about Lincoln, Nebraska, and types in the query "lincoln" (not realizing immediately that this will bring up a lot of information that he is not interested in), it matters little to the searcher that this may make him remember that Lincoln, Nebraska is named after Abraham Lincoln. In fact, this probably is a distraction. Ideally, he would see a list of the categories that his search could take him to, and allow him to refine his search, before it displayed any Web pages at all. Of course, most search engines do not do this (but see Section 6.1.6 for some exceptions), although Google now provides links to the Open Directory (Dmoz) for pages that it knows are in Dmoz, so it shows searchers the categories of the pages that it finds relative to a particular query.

Continuing with "Lincoln" pages, Google also returns pages about the Lincoln Park Zoo in Chicago, another Lincoln University, this one in Canterbury, New Zealand, yet another Lincoln University (in Missouri), the MIT Lincoln Laboratory, Lincoln Memorial University in Tennessee, U.S. Senator Blanche Lincoln, Lincoln Christian College in Lincoln, Illinois, the home page of Lincoln Stein, who is a well-known computer programmer and researcher. And this only takes us through the 32nd of the results returned by Google. While the plurality of pages are about Abraham

Lincoln himself, it is only 12 of the first 32.

The "Ford" query also yields a number of ambiguous results. The first four results returned by Google refer to the Ford Motor Company. The 5th is a link to the Ford Foundation, which is related in origin to the Ford Motor Company. There are a number of links to various foreign dealers and branches of Ford Motor, or related topics, such as a site devoted to racing Fords. There is a site devoted to the Henry Food Museum near Detroit. It is only when we reach links 15-20 in the list that we find links that are totally semantically distinct from Ford Motor; there are pages concerned with Ford's Theatre in Washington, D.C. where Abraham Lincoln was shot, and pages concerned with President Gerald Ford. Following this, are pages concerned with: the Betty Ford Center (for substance abuse), Henry Ford Hospital (in Detroit), Representative Harold Ford of Tennessee, the Ford Amphitheaters in Los Angeles, the actor Harrison Ford, the Gerald Ford School of Public Policy at the University of Michigan, a company called "WJ Ford Surplus Enterprises," a company called "Ford Meter Box," and Ford County, Illinois. While a number of these are quite semantically disjoint (due to the commonness of "Ford" as a family name in the English-speaking countries), there are semantic linkages between somewhat distinct entities, such as the link between Ford Motor and the Ford Foundation, and the link between President Ford and the public policy school named after him. Ironically, despite Google's reputation for being the best search engine on the Internet as of this writing, it—like other search engines—gives the searcher little ability to automatically refine her results, other than allowing the ability to manually form a more detailed query or search within the existing query, as other search engines do.

I downloaded the top 200 results for the "jaguar," "Lincoln," and "Ford" queries

Table 6.1: Characteristics of Pages (URLs) Returned to Three Queries to Two Internet Search Engines; Top 200 URLs Returned by Each Search Engine are Kept

Query	# from Lycos Only	# from Google Only	# from Both	Total # URLs
Jaguar	122	122	78	322
Lincoln	102	102	98	302
Ford	108	108	92	308

from Google and from the competing Lycos search engine. My results indicate that there are substantial differences between the behavior of the various search engines, if these two are representative of the universe of search engines.

For the "jaguar" query, there are 78 URLs in common returned by the two engines. This means that each result set has 122 URLs that are not in the top 200 of the other result (although probably many of them are further down the list), and that the combined sets contain 322 URLs. For the "Lincoln" query, there are 98 URLs in common, so that each has 102 URLs unique to itself, for a total of 302 URLs. For the "Ford" query, there are 92 URLs in common, so that each set has 108 unique URLs, for a total of 308 URLs in the union. These data are summarized in Table 6.1.

Method

For the purposes of computing the connected components, I used the combined results of the two engines, so the sets varied slightly in size (the 322, 302, and 308 union sizes above). There are two ways to compute the connected components. The first way is to consider only direct connections between pages. The second is to allow indirect connections (with one intermediate page). Of course, one could allow even more than one intermediate page, but I have not done this. The number of indirect connections grows exponentially with the distance between pages, making retrieving pages beyond even one indirect connection unwieldy. For instance, the 308 pages in the "Ford" set are connected to an additional 4,117 pages, giving a ratio of an average of 14.3 links per page. Projecting this to the next level of connections would give about 63,000 pages. (Of course, this is ultimately bounded by the total size of the Web and the fact that it is sparsely connected).

Problems with highly-referenced pages connecting unrelated pages emerge only with indirect connections, since all the direct connections are by definition on topic (since all the query results contain the search term) and therefore probably not highly referenced on a global level (that is, with reference to the Web as a whole, such as www.yahoo.com). Including the indirect connections has the advantage of potentially fusing semantically-related sets, at the expense of possibly fusing semantically-unrelated sets, usually through highly-referenced pages.

As we have seen with other relatively small sets of Web pages, the number of direct connections between these sets is quite sparse. Ignoring self-links between pages, there are only 98 direct links (involving two pages) between pages in the "jaguar" set of 322 pages. There are 203 direct links in the "Lincoln" set of 302 pages. There are 80 direct links between the pages in the "Ford" set of 308 pages. These links are very unevenly distributed among the pages in the sets. These data are summarized in Table 6.2.

Thus, in principle, the complexity of computing the connected components is $O(n^2)$ where n is the number of pages involved, because this is the complexity of the number of possible connections between pages, and in order to compute these components, you consider actually connected pages, merging sets when there is a connection between pages that that were, up to that point, believed to have been in

Table 6.2: Number of Direct Links Between Pages in Each Set for Three Queries

Query	Size of Set (n)	Potential Number of Links (w/o Self-Links) ($n(n-1)$)	Actual Number of Links (w/o Self-Links)
Jaguar	322	103,362	98
Lincoln	302	90,902	203
Ford	308	94,556	80

different sets. However, since the connections are actually very sparse, much more sparse than they could be in a completely-connected graph, the performance is much better than this. From the three examples above, we can see in each case that the number of direct links is actually smaller than the number of pages, let alone the square. However, since these are human-created artifacts, one could in principle find sets that were close to completely connected, but in practice this would be quite rare, and such sets would probably be quite small.

Results: Direct Links Only

I used the direct links to group pages together into sets (connected components, ignoring the direction of links). In the case of the "jaguar" pages, there are 17 distinct connected components, composed of 76 of the 322 pages. One of these sets has 29 members, and another has 13 members. One set has 4 members, two have 3, and the rest of the sets consist of only pairs of connected pages. Most of the pages are singletons–that is, they are not connected to any of the others in the set.

For the "Lincoln" pages, there are 20 distinct connected components (sets), composed of 110 of the 302 pages. One of the sets has 48 members. The next largest set has 15 members; after that, there is a set with 7 members, one with 5 members, two with 4 members, one with 3 members. The rest of the sets consist of pairs of pages.

For the "Ford" pages, there are 16 distinct connected components (sets), composed of 69 of the 308 pages. There is one set with 25 members, one with 8 members, one with 6 members, one with 4 members, three with 3 members, and the rest are pairs of pages.

In my judgment, this algorithm does quite a good job of connecting only semantically-related materials, although of course there is no guarantee of this, because anyone can place an unrelated page on the Web and link to a related page. We see that the performance is good by looking through the results of these examples as follows:

In the case of the "jaguar" pages, the set of 29 pages are all about the Jaguar automobile, except for a single page, www.savethejaguar.com, which is a page devoted to the preservation of the Jaguar (animal) as a species. This site is sponsored in part by the Jaguar automobile company, to which it is linked, and therefore it falls into the same set as that site. The set with 13 members are all about the Atari Jaguar video game. All of the other smaller sets contain (within each one) semantically-related pages, about either the animal, the car, the video game, or resorts with "jaguar" in their name. Note that not all of the meanings of "jaguar" that are detailed above appear here, because only a fraction of the pages in the set are directly connected to another page in the set.

For the "Lincoln" pages, the set with 48 members consists entirely of pages about Abraham Lincoln and his family. The set with 15 pages consists of pages about Lincoln, Nebraska. The seven member set consists of pages about Lincoln City on the Oregon Coast, and the surrounding area. The five member set consists of pages about the Lincoln automobile. One of the four member sets consists of pages from the University of Nebraska at Lincoln, and the other consists of pages from Lincoln University in New Zealand. The remaining sets are concerned with topics such as the Lincoln Highway, some more pages about Lincoln, Nebraska, some pages about Lincoln County, Kansas, some about the Lincoln United soccer club in England, some about Lincoln College in Lincoln, Illinois, some about Lincoln University in Missouri, some about Lincoln County, West Virginia, some about the University of Lincolnshire in England, some about Lincoln Center in New York, some from a Web site run by a man named Lincoln Stein, and some about the Lincoln Park Zoo in Chicago. The semantic separation is perfect, but some of the sets (notably the two about Lincoln, Nebraska) need to be merged.

For the "Ford" pages, the set with 25 members are all directly related to the Ford Motor Company or its founder, Henry Ford, except for the "Ford Rugby" page in New Zealand, which features rugby teams sponsored by Ford Motor. The set with 8 members are all related to President Gerald Ford. The 6 member set consists of pages about the actor Harrison Ford. The remaining sets are consist of other pages about Ford cars, the model Patricia Ford, Ford County, Illinois, the Ford Foundation fellowships, Web pages about the small UK company Ford and Mason Ltd, and pages about Ford's Theater, where Lincoln was shot. Each set contains only pages about the topic in question, so again, the semantic separation is quite good.

Table 6.3 summarizes the above data, showing the two largest connected components for each query. As we can see from this table, semantic separation is good for these large components, and we have seen that it is good overall.

Table 6.3: The Two Largest Connected Components in Size (Number of Pages) Found in Each Set of Pages for Each of the Three Queries

Query	Set Rank in Size	Set Size	Set Topic
Jaguar	1	29	Jaguar automobile (all but one)
Jaguar	2	13	Jaguar video game
Lincoln	1	48	Abraham Lincoln (and his family)
Lincoln	2	15	Lincoln, Nebraska
Ford	1	25	Ford Motor & Henry Ford (all but one)
Ford	2	8	Gerald Ford

After the clusters are formed, they could be classified. One way to classify them would be to use the multi-resolution Naive Bayes method in combination with data from Dmoz, which I have described in Section 4.2, assuming that Dmoz's editors have already set up a category for each of the meanings of the query. Each page in each cluster could be assigned to a category with this method. Then the cluster as a whole could be assigned to the category to which the largest number of its members were assigned. Or, it could be assigned to the top two or more categories, depending on how many labels one wants to assign to each cluster.

Results: Both Direct and Indirect Links

I also ran the connected component algorithm using both direct and indirect links. These are links that are mediated through a page not in the set. For instance, if pages A and B are both in the set, and both are connected directly (in either direction) to another page C not in the set, then A and B are considered to be connected indirectly. In this version, A and B are connected if they are connected directly or indirectly. An indirect connection is not counted if page C is among the set of the

5,000 most frequently referenced pages on the Web, as computed by using the sample of one million references taken from Yahoo! and Dmoz as described above.

The "Ford" set, using this methodology, excludes connections via 72 pages (C in the description above), which are the 72 that happen to be linked from the "Ford" set that are in the 5000 most frequently referenced pages. These pages include such well-known pages as www.apple.com, www.google.com, www.microsoft.com, www.whitehouse.gov, and www.netscape.com, as well as some that are not as obvious. However, I was not able to determine that a single one of these globally highly-referenced pages was directly about any of the several meanings of the word "Ford." This is a good thing, because one would not want to break the connection via C if this was the case. Generally, I believe this technique will work because most Web queries are not going to want any of these most-referenced pages as part of their results—the searcher will be looking for something more specific.

This method leads to a modest increase in the number of pages included in the connected components. For the "Jaguar" pages, 96 of the 322 pages are now included, as opposed to 76 when using direct links only. For the "Lincoln" pages, 133 of the 302 pages are included, as opposed to 110 with direct links. For the "Ford" pages, 101 of the 308 pages are included, as opposed to 69 using direct links.

For the "jaguar" pages, there is now one set consisting of 39 pages, one consisting of 19 pages, one consisting of 5 pages, and one consisting of 3 pages. The rest are all pairs of pages.

For the "Lincoln" pages, there is now one set consisting of 54 pages, one consisting of 17 pages, one consisting of 9 pages, two with 5 pages each, one with 4 pages, and 3 with three pages each. The rest are all pairs.

For the “Ford” pages, there is now one set consisting of 32 pages, one consisting of 9 pages, one consisting of 6 pages, two consisting of 4 pages each, and three consisting of 3 pages each. The rest are all pairs.

As in the case of direct links only, the semantic separation is quite good (from my experiments, it would not have been as good if the top 5000 most referenced links had not been excluded). The size of the largest set has increased in each case, in the case of “jaguar” from 29 to 39, in the case of “Lincoln” from 48 to 54, and in the case of “Ford” from 25 to 32. In each case, each of these sets remains almost entirely on the most frequently occurring topic in the sample, which are, as noted above, Jaguar cars, Ford Motor and associated entities, and Abraham Lincoln respectively.

For the “jaguar” pages, the set of 39 pages are all about the Jaguar automobile, except for the same Jaguar (animal) conservation page that showed up in the connected components involving the direct links only. The set of 19 pages are all about the Atari Jaguar video game, an increase of 6 pages over the 13 that appeared among the direct links only. The set with 5 pages are from a resort with “jaguar” in its name, and the set with 3 pages is also about Jaguar automobiles. Most of the rest of the sets are also about the automobile, with a few being about miscellaneous entities that have “Jaguar” in their name.

For the “Lincoln” pages, the set of 54 pages are all about Abraham Lincoln. The set of 17 pages are all about Lincoln, Nebraska, up from 15 using direct links only. The set of 9 pages are about Lincoln, Oregon, up from 7 using direct links. The remaining sets are about all the same topics that were found with the direct links, again with excellent semantic separation.

For the “Ford” pages, the set of 32 pages are all about the Ford Motor Company

Table 6.4: The Two Largest Connected Components in Size (Number of Pages) Found in Each Set of Pages for Each of the Three Queries, Using Direct and Indirect Links

Query	Set Rank	Set Size	Increase over Direct Links Only	Set Topic
Jaguar	1	39	10	Jaguar automobile (all but one)
Jaguar	2	19	6	Jaguar video game
Lincoln	1	54	6	Abraham Lincoln (and his family)
Lincoln	2	17	2	Lincoln, Nebraska
Ford	1	32	7	Ford Motor & Henry Ford (all but one)
Ford	2	9	1	Gerald Ford

or Henry Ford or the Ford Family, except for a page about a Rugby team sponsored by Ford. The set of 9 pages are all about President Gerald Ford. The 6 pages are about the actor Harrison Ford. All the remaining pages are on the same topics that were found using the direct links; the semantic separation is once again excellent.

Table 6.4 summarizes the results from the three queries. The differences from direct links only, as previously shown in Table 6.3, are noted.

Discussion

The above technique of finding connected components after pruning out the most-globally-referenced pages, is very effective in separating groups of Web pages that are the result of a search engine query into semantically-related components. We have seen this for three distinct Web queries. The version of the technique that uses both direct and indirect (two-hop) links is more effective than the technique that uses direct links only.

Both versions of this technique are quite tractable computationally. In practice,

Table 6.5: Set Size, Direct Internal Links, and Total Link Count for the "Ford," "Jaguar," and "Lincoln" Sets of Pages

Query	# Pages	Direct Links	Total Out-Links	Mean Out-Links/Page
Ford	308	80	5233	17.0
Jaguar	322	98	6714	20.1
Lincoln	302	203	6355	21.0

this technique would be used in conjunction with a search engine. Search engines tend to keep track of links as they retrieve them, in order to spider to links that they have not seen before. Pulling links out of html files is a relatively cheap activity, as html files (which are text only) tend to be quite small (as opposed to image or multimedia files). Also, the number of links on a page tends to be quite tractable. Table 6.5 shows the number of pages, direct links between these pages, and total out-links to all pages for the three sets of pages we have studied above. In forming the sets, even in the more taxing task of using indirect links, the system only needs to visit each of about 20 links per page, on average, and only needs to visit this linkage once in forming the connected components, because the algorithm, as described in Figure 6.5, does not require any backtracking or repeat iterations through the set of linkages. Thus I believe such an algorithm could be deployed in real-time on a public search engine without the consumption of excess resources. In addition, connected components could be computed in advance of any actual queries.

One flaw in this technique is that it does not group together pages that happen to be on the same topic but happen also not to be linked on the Web. This technique could be used as an initial phase in clustering; textual comparison of the various components with one another could be used in order to determine whether components should be merged.

6.1.5 Finding the Most Salient Pages within Connected Components using In-Link Counts

Hypothesis

The connected components described above can also be used to determine the possibly most significant pages within each connected component of a set of pages.

Method

Within each component, I find the pages that are pointed to by the largest number of other pages in the component. The technique is outlined in Figure 6.8. This is a somewhat similar technique as that used by Google (except that it is not recursive). However, it has the advantage over the technique used by Google in that all the pages in each connected component are likely to be on a single topic. Therefore, the highly ranked pages in each connected component, as found by a count of such in-links, are likely to have both high relevance to a particular topic represented by a clique on the Web and also high quality or salience within that topic.

This is illustrated by the hypothetical 5-page connected component shown in Figure 6.7. In this figure, the node (page) labelled A has the largest number of in-links (4), so it is selected as the most significant page in the component.

Figure 6.7: A Hypothetical 5-Page Connected Component.

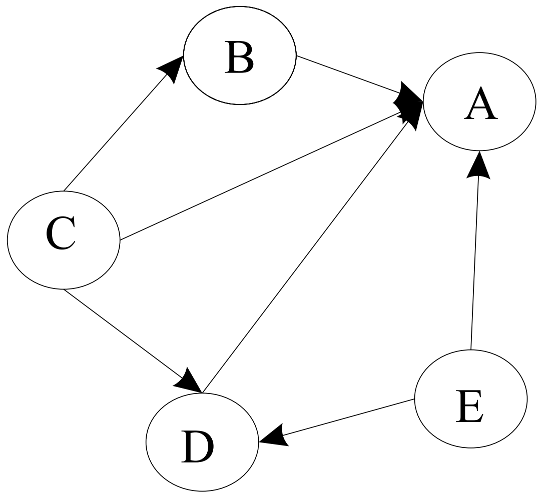


Figure 6.8: Finding the Pages in Each Connected Component of a Set of Web Pages W with the Highest Number of In-Links

1. Compute the connected components of W as described in Figure 6.6.
2. For each component C of W:
   (a) For each page P in C, count how many other pages in C also point to P. Call this count *inlinks(P,C)*.
3. For each component C of W, place each page P in C in descending order of *inlinks(P,C)*. It is likely that the most salient pages in each component have the highest values of *inlinks(P,C)*.

Results

For the “jaguar” query, using only direct in-links, the largest connected component is about the Jaguar automobile, and contains 29 pages, as we have seen. The page with the largest number of in-links within this set has seven in-links, and is www.jaguar.com, the Jaguar cars home page. The second largest number of in-links is a page for the Jaguar owners Web ring; other member pages in this ring link to that home page of the ring.

The second largest connected component has 13 members, is about the Atari Jaguar video game, and has three pages tied for the most number of in-links (3 each). All three of these are pages devoted to the game built by devotees, and all are highly salient.

For the “Ford” query, again considering solely direct in-links, the page with the highest number of in-links in the (largest) connected component of 25 pages, which are largely about Ford Motor, Ford automobiles, and Henry Ford, is www.ford.com, the Ford Motor Company home page, which has 15 in-links. The second largest number is www.fordvehicles.com, another official Ford Motor Company page, with seven in-links. (The number of total in-links remains quite sparse; for instance, this component of 25 pages could have potentially $n(n-1) = 25 * 24 = 600$ directed in-links, but it only has 39 in total.)

Since these prominent Ford Motor company pages were found, this technique appears to do a good job in identifying the most salient pages within the component. In the second largest component, which contains eight pages and is about Gerald Ford, the page with the most in-links, 3, is the official page of the Gerald Ford Library and Museum, again, a highly salient page.

For the "Lincoln" query, again utilizing only direct in-links, the page with the highest number of in-links, eight, in the largest connected component of 48 pages about Abraham Lincoln is a site called the Abraham Lincoln Research Site. Following this in close competition is a page with seven in-links is a highly detailed page about Lincoln's assassination which has won a number of high reviews and site awards. (These pages were both developed by the same author and illustrate a vulnerability of simply counting in-links, since pages controlled by the same person or organization can link to one another and boost their scores.) For the next largest connected component, 15 pages which are about Lincoln, Nebraska, the two most referenced pages within the component each have four in-links. One of these is the city's daily newspaper's page, and the other is the home page of the city and county governments. These are both obviously highly salient to the topic.

Thus we have seen that in all cases we have examined, the technique of identifying high in-link pages within each component finds highly salient pages.

6.1.6 Looking at the New Search Engines: Teoma and Wisenut

In late 2001, at least two new search engines emerged that combined subject-relevant link information (communities or clusters of pages) with in-link-counting, as used by Google and Science Citation Index. These were Teoma (at www.teoma.com) and Wisenut (at www.wisenut.com). I tested both of these on the three queries above. Despite the usual hype, both systems missed obvious topics that were detected by the simple connected component technique described above. And as is unfortunately typical with many such systems, the systems are quite proprietary and secretive about the techniques employed.

For the Lincoln query, Wisenut got most of the major meanings—Abraham Lincoln, the Lincoln automobile, and Lincoln, Nebraska, but fouled up with respect to "Lincoln City", mixing up cities that were widely dispersed (Nebraska, England, Oregon), and also malfunctioned with respect to "Lincoln County" in a similar manner. There was no identified group of pages about Lincoln City and County, Oregon, even though I found a component on the Web of such pages. It had some categories that are lacking above, such as one for the Lincoln Trail (but this is probably because I have quite a limited sample of Web pages), but was also lacking some other categories that I found (e.g. the Lincoln Park Zoo). A technique that labeled the connected components that I identify above by the most frequently occurring word pairs within them, after dropping words off of a stop list, would work better, than whatever technique Wisenut is using. Wisenut's system is also quite slow, much slower than Google, at least for the queries I tried, although this may be a function of server or network load rather than algorithmic design.

For the Lincoln query, Teoma gives fewer categories than Wisenut. Unlike Wisenut, it did not make the mistake about Lincoln City and County. But it totally missed quite a few topics, such as Lincoln Center and the Lincoln Park Zoo. It mis-labeled its set of pages about the Lincoln Automobile as "quality care," presumably because many of the Ford Motor Company sites use this phrase, and it uses some sort of common phrase extraction algorithm to label the sets. Teoma's performance is better than Wisenut's.

For the Ford query, Wisenut did a good job of finding most of the major meanings, although it missed some of the minor ones, such as pages about the model Patricia Ford and about Ford County, Illinois. Frankly, I am mystified why it missed these—

Table 6.6: Separation/Detection of Four Meanings for Each of Three Queries by Teoma and Wisenut

		Detected/Separated by:	
Query	Meaning	Teoma	Wisenut
Lincoln	Abraham Lincoln	Yes	Yes
Lincoln	Lincoln Automobile	Yes	Yes
Lincoln	Lincoln Nebraska	Yes	Yes
Lincoln	Lincoln Oregon	No	No
Ford	Henry Ford	No	Yes
Ford	Ford Motor Company	Yes	Yes
Ford	Gerald Ford	Yes	Yes
Ford	Harrison Ford	No	Yes
Jaguar	Atari Jaguar	Yes	Yes
Jaguar	Jaguar Automobile	Yes	Yes
Jaguar	Jaguar (animal)	No	Yes
Jaguar	Jacksonville Jaguars	Yes	No
Total Percent Detected/Separated:		66.6%	83.3%

the only explanation I can think of is that its page set is still too small, although this is strange, because I picked these up with a small page sample. However, that may have been drawn from a larger index. For the Ford query, Teoma missed all the major topics except for those related to the Ford Motor Company and Gerald Ford; it missed Henry Ford, Harrison Ford, and the Ford Foundation, for example. This seemed to me to be quite poor performance.

Both Wisenut and Teoma have the disadvantage of classifying at only one level of resolution, a flaw that would not be shared by a Dmoz-based system. For this example (Ford), both systems list multiple topics that should have been organized under a single topic for Ford automobiles (e.g. after-market parts, Ford Falcon, etc.)

For the Jaguar query, Wisenut found pages related to the Atari Jaguar and to the automobile. It also found just two pages related to the animal. It missed some of the subsidiary meanings, such as the Jacksonville Jaguar football team and the chemistry software named Jaguar. Teoma found the meanings for the automobile, for the Atari Jaguar, and for the Jacksonville Jaguars, but missed the animal meaning and the chemistry software.

Table 6.6 shows that Teoma and Wisenut do a pretty good job of picking up and separating four major meanings of each of the three queries studied; Wisenut does a better job than Teoma. As we have seen, where these systems fail is on some of the less common meanings, which is what you might expect, since these less common meanings offer less data on the Web with which to detect patterns.

Neither of these systems appear to work better than Northern Light, which has been doing something quite similar for a longer period of time, but without the "buzz" in the community. However, Northern Light's Web search engine is no longer publicly available.[8]

6.1.7 Directions for Further Research

It is likely that the results of of the clustering that results from computing the connected components could be successfully refined using a conventional document-similarity-based method such as a partitional clustering method, such as that used in [56]. Instead of selecting the cluster seeds randomly, and building initial clusters around them, the connected components could be used as the initial clusters, and iterations could then be used, reassigning documents that fit better in a different

[8]Northern Light has decided to refocus their business on doing searches within proprietary sources and doing specialized searching for corporate and other institutional clients.

cluster, and perhaps eliminating some of the smaller clusters altogether. This could fix the problem, associated with the connected component method, of two or more components each consisting of pages that are on the same topic but just happen not to be connected. This occurs frequently with commercial sites on the same topic, since they compete and therefore do not inter-link [119]. It also happens often simply because Web authors do not add links, due to a lack of knowledge, interest, or time.

Another approach would be to incorporate the link information directly into the document similarity function, which then could be used with any clustering algorithm. Text similarity and the hyper-link distance between documents are both ways of estimating the underlying semantic similarity between documents. A method that estimates this similarity using both of these is likely to perform better than one using either one alone. Since document similarity measures are basic to all document clustering methods, a better similarity measure is likely to lead to a better clustering outcome. The following is one way to design such a hybrid clustering measure.

Let

$$sim_T(d_i, d_j)$$

be the similarity of the text of two documents, used for text clustering with a variety of algorithms (see [106, 95]). Usually this is the standard term vector cosine similarity [5], although alternates (e.g. [41]) may perform better. Then

$$sim_{TN}(d_i, d_j) = \frac{sim_T(d_i, d_j)}{\sum_{k \in D, l \in D} sim_T(d_k, d_l)}$$

is the normalized similarity, where D is the set of all documents.

Let

$$sim_L(d_i, d_j) = \frac{\left(\frac{1}{udist(d_i,d_j)} + \frac{1}{ddist(d_i,d_j)} + \frac{1}{ddist(d_j,d_i)}\right)}{3}$$

and

$$sim_{LN}(d_i, d_j) = \frac{sim_L(d_i, d_j)}{\sum_{k \in D, l \in D} sim_L(d_k, d_l)}$$

where $udist(d_i, d_j)$ is the link distance between d_i and d_j with the links viewed as undirected, $ddist(d_i, d_j)$ is the distance from d_i to d_j along directed links, and $ddist(d_j, d_i)$ is the directed distance in the other direction. Note that the subscripts L, T, and N refer to "links," "text," and "normalized" respectively. Note that any of these is set to be infinite, if the respective undirected or directed path does not exist. Also, high frequency pages such as www.netscape.com or www.yahoo.com may be pruned out before finding these paths so as to have these distances be more closely related to the semantic distance between pages.

Let

$$sim_{TLN}(d_i, d_j) = \frac{sim_{TN}(d_i, d_j) + sim_{LN}(d_i, d_j)}{2}$$

be a combined similarity value. Note that it is normalized because it is the average of two normalized values. This similarity measure treats link structure and text information with equal weight, and could be used in experiments to see whether in fact link information improves clustering performance.

6.2 Using WordNet to Disambiguate Results

WordNet [35] (described in Section 2.3) contains enormous quantities of semantic information, assembled laboriously by trained lexicographers and linguists. However, it does not contain a lot of information about proper names and the specific entities associated with them (since this would be closer to the purview of an encyclopedia rather than a dictionary, and WordNet is closer to a dictionary, albeit a very sophisticated one.)

For the three examples given above ("Ford," "Lincoln," and "jaguar"), WordNet can only be of partial assistance in disambiguating the results. It "knows" (that is, has information in its database) the most about "Ford," listing President Gerald Ford, Henry Ford, and several other well-known people, as well as well as the underlying sense of a crossing of a stream or river. However, it does not have any listing for the Ford Motor Company, which is the topic of the largest group of pages returned by the search engine. For "Lincoln," it knows about Abraham Lincoln (the most frequent page topic) as well as Lincoln, Nebraska, but none of the other senses listed above. And for "jaguar," the situation is the worst—WordNet knows only about the animal, but none of the animal's namesakes, including the automobile. So, since it only knows about one meaning, it would be useless in disambiguation. It is difficult for a relatively small[9] team of lexicographers to keep up with the wide range of meanings used in the real world.

6.2.1 Hypothesis

WordNet can be a useful tool in disambiguating search engine results.

[9]Small, that is, relative to the vast universe of English speakers.

Figure 6.9: Disambiguation of Web Page Results on a Query for a Specific Word X Using WordNet

1. Find all the meanings of the word X in WordNet
2. For each meaning M of X, concentanate the synset (set of synonyms), the gloss (dictionary definition) and the usage examples, and remove words on the stop list. Transform this concatenation of words to a normalized term vector, as described in Section 2.7. Call this vector m.
3. Find all the Web pages returned by a search engine in response to a query of X.
4. Extract the text from each Web page, and transform it to a normalized term vector. Call this vector w.
5. Consider all dot products $m \cdot w$. For each W, assign it to the meaning M for which it has the highest value of $m \cdot w$.

6.2.2 Method

A simple disambiguation algorithm using WordNet would be the following: Consider a search for a word X and the results set returned by a search engine from a query of X. Then consider all the meanings of X in WordNet. Take the synset (set of synonyms), the gloss (dictionary definition), and the usage examples given for each meaning in WordNet and transform the concatenation of these three things into into a normalized term vector (after removing words on the stop list). Then do the same for each of Web pages that are returned by the search. For each page, classify it with that meaning with which it has the highest dot product (vector similarity). Compare the results with a manual classification to determine performance. This disambiguation technique is outlined in pseudocode in Figure 6.9.

6.2.3 Results on the Ambiguous Query "Speaker"

Because WordNet does not have data about the major meanings of the three words that we have been working with up until now, I decided to test it with a common word for which it would have some of the major meanings. The word "speaker" has three meanings in WordNet, which all derive from the underlying meaning of "to speak" in different contexts. They are:

1. speaker, talker, utterer, verbalizer, verbaliser – (someone who expresses in language; someone who talks (especially someone who delivers a public speech or someone especially garrulous); "the speaker at commencement"; "an utterer of useful maxims")

2. loudspeaker, speaker, speaker unit, loudspeaker system, speaker system – ("electroacoustic transducer that converts electrical signals into sounds loud enough to be heard at a distance")

3. Speaker – (the presiding officer of a deliberative assembly; "the leader of the majority party is the Speaker of the House of Representatives")

The above three definitions are WordNet's actual output for the dictionary query "speaker," consisting of the synset (set of synonyms) and the gloss (definition); the gloss may, in some cases, contain usage examples. However, even these three meanings do not cover all of the meanings of the word "speaker" found on the Web. A Google search for the word "speaker" returns about 4.7 million results. The first result is for a newspaper which is called the "Standard Speaker" and therefore does not really fit into any of the above three meanings of the word. The second and third results are for loudspeakers and the Speaker of the House in Texas, but the fourth result is a page about a text-to-speech computer program called Speaker. The twelfth result is for another newspaper with "Speaker" in its name.

I used the Lycos search engine for this experiment.[10] Lycos gave a similar mix of pages, in terms of their distribution among the various meanings of "speaker." I classified the pages manually. Of the first 100 pages returned by Lycos, over half (51) are pages about loudspeakers (there is a lot of interest in loudspeakers on the part of Internet users). 30 pages refer to human speakers of one sort or other (other than "Speakers of the House"); most of these are home pages of motivational speakers or of speakers' agents or bureaus. Six pages were associated with Speakers of the House, on either the federal or the state level. Finally, 13 pages did not fit into any of these categories. A batch retrieval program was able to retrieve 97 of the 100 pages; of the three that were not retrieved, two were among the 13 that did not fit any of the three meanings, and one was in the "human speaker" meaning (meaning #1).

I applied the WordNet-based algorithm described in Figure 6.9. Disregarding the 13 pages that did not correspond to one of the 3 meanings, and the one other page that did not retrieve, this leaves a total of 86 pages. By chance, classification into 3 categories should get one-third of these, or almost 29, correct. In fact, this algorithm got 58 correct, or about 67 percent of those pages that were known to be in one of the three categories before automatic classification, about twice what it could be expected to get by chance. The confusion matrix for the three meanings is given in Table 6.7.

[10]On the particular day I was searching, my batch program was not working well in retrieving pages from Google, so I switched to Lycos. For a task like this, there is no particular reason, that I know of, to choose one search engine over another.

Table 6.7: Confusion matrix for WordNet-based classification of three meanings of the word "speaker"

		Automatic Classification		
Manual Classification	total	human speakers	loudspeaker	Speaker of the House
human speakers	30	8	13	9
loudspeakers	51	4	45	2
Speaker of the House	6	0	1	5

I did three searches on Google to attempt to disambiguate results: the search strings were: "speaker public speech," "loudspeaker," and "speaker house." These strings were drawn from words in the three WordNet meanings, using my own judgment. In all three cases, in my judgment, the vast majority of the pages returned were reflective of that meaning. This indicates that WordNet can be quite useful in query refinement, if a proper interface is built to allow the user to select additional terms. However, one cannot simply throw all the words in a definition at a search engine, because the search engines typically use a brittle (all-or-nothing) rather than a fuzzy logic [129] to match pages against word strings. And the success of the query refinement would depend heavily on the sophistication of the user in selecting additional terms. As we have already seen, it also depends heavily on WordNet having the meaning of the original term that the searcher had in mind in its database. WordNet would be a more useful tool if a good deal of additional energy was invested into making it closer to an encyclopedia rather than simply a dictionary, and if it was made to contain much information about culture and about proper names.

Note that the level of polysemy of a word tends to be directly related to its commonness in the language. Thus common, short words such as "line" and "hard" have a lot of meanings in WordNet; "line" has 29 meanings as a noun and seven as a verb; "hard" has 11 meanings as an adjective and ten as an adverb. Many of these meanings are close to one another; for instance "hard" can mean "vigorous" or it can mean "arduous"; here the difference is only one of degree. So the process of disambiguation becomes difficult and fuzzy [78].

6.2.4 Results on the Ambiguous Query "Bear"

Another example of an ambiguous word is the word "bear." According to WordNet, "bear" has two meanings as a noun and thirteen meanings as a verb. As a noun, "bear" can refer to the carnivorous animal or the investor in the stock market who believes that prices will decline. As a verb, two of the meanings of "bear" are to give birth ("bear a child") and to carry ("bear a load").

Here, we will be concerned with disambiguation of the two noun meanings stated above (as we will see below, there are other metaphorical noun meanings that WordNet does not list; one of the main problems in AI and cognitive science, is in dealing with such metaphorical use of language, which is highly prevalent, see for example, [62][11]), ignoring all of the verb meanings. However, we will use a somewhat elaborated method of disambiguation than we used with the example of "speaker" above.

We will not consider just the synonym set, gloss, and usage examples associated with the meaning of a word, but in addition, its semantic neighborhood. Here, I define

[11] Lakoff and Johnson [62] point out that the use of metaphor is highly prevalent in language and structures the way we think about language. For instance, we use metaphors from war to describe arguments; e.g. "I demolished his argument." This makes it difficult for automated systems that have at best a surface knowledge of language, on the level of word correlations, to do a very good job of disambiguation, because the machine can easily mistake a metaphorical usage for a non-metaphorical one. An understanding of the deeper meaning of the metaphor would be necessary to have the machine truly understand the context of such language use.

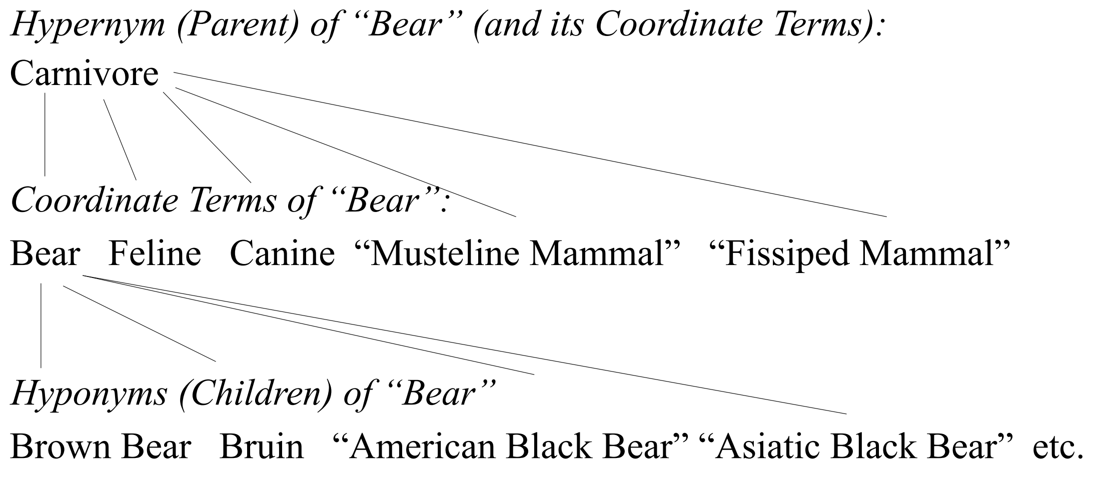
the semantic neighborhood of a noun as its immediate hyponyms, hypernyms, and coordinate terms (the latter are “sisters”; that is, other children of their immediate hypernym or parent). I am still using the same algorithm described in Figure 6.9. However, I am adding to the word stems mentioned in step 2 of that figure all of the word stems from the semantic neighorhood so defined.

I could define the semantic neighborhood more widely if I chose, for instance, going further up or down the chain of specificity and including more coordinate terms at each level, or by following keywords in the definitions for other words related to them, but I did not do that in this experiment, to keep it relatively simple.

For the sense of “bear” as a carnivorous animal, its immediate hypernym (“parent”) is the word “carnivore.” Its immediate hyponyms (“children”) includes a list of specific bears, including the brown bear, the bruin, the American black bear, the Asiatic black bear, the polar bear, and the sloth bear. Its coordinate terms are other types of carnivores, including canines, felines, and some other more esoteric ones, such as “musteline mammal” and “fissiped mammal.” This WordNet neighborhood is illustrated in Figure 6.10. The definition given by WordNet is the following: “massive plantigrade carnivorous or omnivorous mammals with long shaggy coats and strong claws.”

For the sense of pessimistic investor, its hypernym is “investor.” WordNet knows about no immediate hyponyms (this is already quite a specific term). Its coordinate terms are terms that are also investors, of different types. WordNet gives the following: “contrarian,” “bondholder,” “bull,” “caller,” “depositor,” “lender,” “rentier,” “loaner,” “stockholder,” “shareholder,” “shareowner.” Some of these terms are synonyms of one another, for instance, the last three listed. The antonym of “bear”—“bull”—is also a

Figure 6.10: A Portion of WordNet Around the Word "Bear," for its Animal Sense


Hypernym (Parent) of "Bear" (and its Coordinate Terms):
Carnivore
Coordinate Terms of "Bear":
Bear Feline Canine "Musteline Mammal" "Fissiped Mammal"
Hyponyms (Children) of "Bear"
Brown Bear Bruin "American Black Bear" "Asiatic Black Bear" etc.

coordinate term. The definition given by WordNet is the following: "an investor with a pessimistic market outlook."

Thus, if we consider the two semantic neighborhoods constructed from these two meanings, we can anticipate that they would do quite a good job in distinguishing the two meanings. For each term in each semantic neighborhood, it appears much more likely that that term would appear on pages that are about that particular meaning rather than the other one. For instance, the word "bondholder" would be more likely to appear on a page that uses the word "bear" in the context of investing; the phrase "polar bear" and the word "polar" would be more likely to appear on pages about bears in the context of the animal. Of course, since the investor meaning of the word is metaphorical and is probably drawn from the proverb "selling the skin before you have caught the bear" [34], one is likely to see some "cross-talk" between the two meanings, as we saw in the case of the metaphorical uses of the word "jaguar." For instance, a bear market might be described as a man-eating carnivore.

I did a Web search on Lycos for "bear" and retrieved the top 300 results. It turns out that the vast majority of these results are not about either of the meanings listed above. The single largest number of pages were about Teddy bears or children's cartoon bears. While in a sense these are bears, I did not classify them as such. There are also a lot pages about places named after bears, such as towns, ski resorts, or various companies named after bears. Apparently, some large, hirsute gay men consider themselves bears (in a metaphorical sense) and there are a fair number of pages devoted to this. There are a fair number of pages that show up because they are about the right to bear arms. There are relatively few pages about the animal—definition one—and even fewer about the investor—definition two. This is another example of how when you search, you seldom find what you think you will. During my manual classification, I classified all pages that were not directly about the animal or the investor into a third category—"other."

Of the 300 search results, only 20 are about the animal, and only five are about the investor meaning. It is not because these meanings are not prevalent on the Web—in fact, more focused searches (such as "black bear," "grizzly bear," or "bear market") do a much better job of pulling out plenty of pages that are on these topics. It is just that they are drowned out by the other meanings when the simple query "bear" is given.

I followed Lycos's 300 hyperlinks, and was able to retrieve the Web pages for all but 21 of them, for a total of 279. The vast majority of these were classified in the animal class by the WordNet-based semantic neighborhood algorithm, as we can see from the confusion matrix shown in Table 6.8.

Here, the separation of the second meaning, "investor," is not terrific. Of course,

Table 6.8: Confusion matrix for Classification of two noun meanings of the word "bear"

		Automatic Classification	
Manual Classification	Total	animal	investor
animal	19	19	0
investor	5	2	3
other	255	247	8

the sample is very small, which is part of the problem. The other part of it, I discovered by looking at the pages in question, is that use of metaphorical language (referring to the animal; e.g. the phrase "the bear devoured the market") on pages actually about investors is as prevalent as it is on other pages that are about metaphorical bears. Also, the strength of the word "bear" in the vector representing the semantic neighborhood for the animal is particularly strong, because the word "bear" occurs multiple times in this neighborhood as different kinds of bears are listed. One way to counteract this might be to allow each word to be represented equally in this vector, instead of according to its frequency in the WordNet text of the semantic neighborhood.

It turns out if you do this—have each term represented equally in the vector—4 out of the 5 of the pages about investors are correctly classified, and for the remaining one, the animal meaning edges out the investor meaning only barely. However, this comes at the expense of mis-classifying 8 of the 19 pages about the animal as investor pages, so this is not a desirable improvement on the algorithm, at least in this case; too high a price to pay to classify just one more page correctly.

It turns out that the main reason for this mis-classification is that the word "bear" is no longer as heavily weighted in the term vector representing the semantic neighborhood for the animal, since all terms are now equally weighted, and it is more heavily weighted in the vector representing the semantic neighborhood for the investor, because there happen to be, in this case, fewer terms in the latter neighborhood (and the vectors are normalized). This results in a greater dot product for those pages that have the word "bear" appearing frequently, which are obviously quite a large fraction of the 300 pages, given that "bear" was the search query. It might be possible to counteract this by counting each term on each page just once for the purposes of forming the term vector, but I did not try this.

6.2.5 Discussion

Adjacent word pairs can be useful in separating out different meanings of words where single words fail, and can be particularly helpful in a scenario where the desired meanings are relatively sparse in the overall search results. A program to extract the adjacent word pairs from the 300 pages, ignoring words on the stop list, resulted in slightly over 30,000 two-word combinations. If you consider only the word combinations that contain the word "bear," there are still over 1,500 such word pairs.

WordNet was not specifically designed for doing disambiguation/separation; it might be a useful project to create sets of keywords or keyword pairs that specifically characterize sets of pages on the Web, although this would be a project of a similar difficulty as WordNet itself. Alternately, WordNet itself could be augmented with such keywords or word pairs. Word pairs or short phrases would probably be best because they tend to be less ambiguous than single words. Thus, the entry for the animal "bear" could be augmented by such word pairs or phrases.

However, in a sense, WordNet already contains such phrases, and could be used as an interface to search. In the first step of the search, one would use it like a dictionary, and it would list each of the meanings of the word. The searcher would then select a meaning, and WordNet would find all of the hyponyms of that meaning. It would then use a fuzzy disjunction of those hyponyms to find pages referring to that meaning. In the case of the animal meaning of "bear," such hyponyms are "black bear," "polar bear," "sloth bear," etc. A disjunctive search for any or all of these meanings will work quite well, especially if those pages that have as many of these phrases as possible would be listed first.

Of course, the searcher would have to pay the extra price of working through a guided search, and many searchers have little patience for such things. Also, the implementation of the search engine would have to be different than that the search engines of which I am aware. This because it would have to be based on a scoring system where pages are ranked higher based on the number and frequency of the presence of the various keywords or keyphrases; that is, a page gets a higher score if it contains more of the keyphrases, in a higher frequency. A simple disjunction will not work; for instance, the disjunction used by AltaVista does not rank the results in this manner; instead, it lists pages that only have one of the items in the disjunction first, inexplicably. Some other search engines do not offer true disjunctive search at all.

6.2.6 The Use of Two-Word Combinations Involving "Bear"

We can try to use these two-word combinations involving "bear" to try to separate out the 300 results we found above. This is a partial test of the above proposal to use the

WordNet hyponyms, since many of the keyword pairs are similar to the hyponyms, or at least to concepts that are associated with the central one. The top fifteen keyword pairs involving the word “bear” are: “big bear,” “Teddy bear,” “grizzly bear,” “bear arms,” “bear lake,” “Smokey bear,” “bear valley,” “black bear,” “bear mountain,” “arm bear,” “billi bear,” “bull bear,” “dancing bear,” and “bear recovery.” Only three of these: “black bear,” “grizzly bear,” and “bear recovery” directly reflect the animal meaning of the word “bear.” Only one, “bull bear,” reflects the investor meaning. (Note that since stop words are eliminated before the computation of the word pairs, phrases like “bull and bear” can lead to “bull” and “bear” being considered adjacent, since “and” is on the stop list.)

“Grizzly bear” is present in seven of the 20 pages about the animal, and one page that is not. “Black bear” is present in five of the 20 pages, and five that are not. In three pages, both words are present. Thus both words together will cover 9 pages. Here we have another familiar example of the often (but not always) occurring recall/precision trade-off; including more of such (two word) phrases in the disjunction often increases recall at the expense of precision, since a wider net is also cast.

Considering the investor meaning: the two word phrase “bull bear” was found on two pages of the five that were about this meaning, and one page that was not. The two word phrase “bear market,” which appeared to be the next most common two word phrase that was on the investor topic, was also found on two of the five; a different two. Thus again, we have increased recall by using a disjunction. Here, however, luckily, precision is also not harmed; in fact, it is improved.

6.3 Using Semantic Networks for Clustering

A HAL-style semantic network [17] (see Section 2.9.2 for a description of such networks) constructed out of a set of Web pages that are the response to a search engine query can be used to refine this query, in order to make it less ambiguous. Techniques for doing so are detailed in this section.

6.3.1 A HAL Matrix for "Jaguar"

Tables 6.9 through 6.12 show one dimension of such a HAL-style matrix, centered on the word "jaguar", constructed from Web pages that are the result of a query on that word.

The HAL matrix shown in Tables 6.9 through 6.12 does not eliminate "stop words;" doing so would reduce noise, at the possible expense of removing salient stop words that are particularly related to a specific query. Eliminating stop words would get rid of quite a few of the words in Tables 6.9 through 6.12, but I have left them in because the HAL algorithm does not eliminate them. However, as we will see below, it may be profitable to eliminate them for further analysis.

This list of words indicates (as did the exploration of the Web pages through cluster analysis of the link structure), that the pages are not equally distributed on the various topics. Most of the words in Tables 6.9 through 6.12 are related to the automotive meaning of the word "jaguar;" the second largest number are related to the video game meaning, and the smallest number, ironically, to the original zoological meaning of the word.

Table 6.9: One Dimension of the HAL-Style Matrix Relative to the Word “Jaguar” (Words 1-50)

rank	word	score	*rank*	word	score
1	jaguar	20637	*26*	system	1444
2	type	6270	*27*	usa	1440
3	game	5407	*28*	sale	1304
4	club	5185	*29*	classic	1301
5	the	4039	*30*	links	1293
6	cars	3898	*31*	dinky	1285
7	part	3767	*32*	driver	1263
8	xj	3167	*33*	bit	1236
9	atari	2583	*34*	howard	1202
10	site	2530	*35*	control	1143
11	volvo	2442	*36*	it	1110
12	models	2405	*37*	series	1103
13	web	2385	*38*	rings	1099
14	dealership	2310	*39*	we	1087
15	news	2141	*40*	home	1058
16	owners	1984	*41*	unit	1046
17	developing	1894	*42*	here	1013
18	world	1797	*43*	this	987
19	server	1622	*44*	racing	985
20	click	1568	*45*	power	952
21	xk	1542	*46*	sedan	943
22	rates	1537	*47*	informative	942
23	cd	1495	*48*	lynx	939
24	pages	1491	*49*	jag	936
25	enthusiast	1457	*50*	corgi	934

Table 6.10: One Dimension of the HAL-Style Matrix Relative to the Word "Jaguar" (Words 51-100)

rank	word	score	*rank*	word	score
51	luxury	911	*76*	including	692
52	related	905	*77*	checks	671
53	association	902	*78*	see	667
54	converter	897	*79*	times	655
55	tour	865	*80*	programs	650
56	york	863	*81*	interactive	648
57	logo	858	*82*	played	642
58	collector	852	*83*	order	638
59	manhattan	844	*84*	support	637
60	orloff	836	*85*	ii	631
61	skunk	819	*86*	forum	630
62	daimler	813	*87*	coupe	628
63	special	803	*88*	nyc	619
64	community	798	*89*	pre	617
65	radiator	778	*90*	get	617
66	specialist	766	*91*	year	616
67	all	761	*92*	use	598
68	automobilia	754	*93*	list	597
69	find	752	*94*	uks	596
70	scale	751	*95*	owned	584
71	in	747	*96*	there	581
72	xke	747	*97*	release	578
73	mk	709	*98*	codes	572
74	for	708	*99*	opinions	571
75	design	708	*100*	cartridge	569

Table 6.11: One Dimension of the HAL-Style Matrix Relative to the Word “Jaguar” (Words 101-150)

rank	word	score	*rank*	word	score
101	cat	564	*126*	reviews	493
102	peripherals	559	*127*	speed	491
103	top	558	*128*	title	487
104	working	555	*129*	mail	485
105	faq	551	*130*	people	483
106	photo	550	*131*	automobile	483
107	inline	541	*132*	provided	482
108	jazzmaster	536	*133*	ranges	481
109	one	535	*134*	performance	479
110	vr	535	*135*	restoring	476
111	updates	527	*136*	names	470
112	details	525	*137*	video	466
113	make	516	*138*	quick	465
114	discussion	513	*139*	bytecode	462
115	wanted	511	*140*	project	461
116	collection	510	*141*	service	457
117	register	510	*142*	spotted	456
118	console	507	*143*	section	452
119	engine	506	*144*	vote	447
120	software	506	*145*	sold	444
121	visiting	500	*146*	made	443
122	source	497	*147*	you	441
123	main	495	*148*	quote	432
124	mr	494	*149*	history	431
125	diecast	494	*150*	official	427

Table 6.12: One Dimension of the HAL-Style Matrix Relative to the Word "Jaguar" (Words 151-200)

rank	word	score	*rank*	word	score
151	virtual	426	*176*	books	384
152	front	425	*177*	coat	372
153	programmers	425	*178*	wedding	372
154	line	422	*179*	group	372
155	cheats	421	*180*	re	371
156	port	421	*181*	members	371
157	hardware	420	*182*	machines	369
158	high	418	*183*	full	367
159	sovereign	414	*184*	last	367
160	runs	414	*185*	hunt	365
161	features	414	*186*	animals	364
162	magazine	411	*187*	north	360
163	websites	409	*188*	edition	360
164	america	408	*189*	these	358
165	network	407	*190*	our	358
166	conversions	406	*191*	coventry	356
167	info	400	*192*	ibm	355
168	typ	397	*193*	computers	353
169	board	397	*194*	architecture	352
170	company	395	*195*	jungle	352
171	ltd	395	*196*	original	352
172	leopards	392	*197*	lair	352
173	pictured	391	*198*	what	352
174	joined	389	*199*	purchase	349
175	iii	385	*200*	*longer*	348

6.3.2 Finding Word Correlates for Use in Query Refinement

The HAL matrix allows one to examine the top correlates of each word. In Tables 6.9 through 6.12, the top word that is involved with the automotive meaning of the word is "type," because some Jaguar automobile models are named "S-type," "X-type" etc. The top word that is involved with the video game meaning is "game." The top word that is involved in the animal meaning is "cat," but this is only the 101st word on the list.

Generation of such a HAL-style list of query word correlates may assist the user in query refinement. If the list of m such words is displayed to the user, she may select, using check-boxes, those n (of the m) words which would assist in refining the query. Then all the selected words would be combined with the initial query word in order to refine the query. I have not implemented such a system yet. However, one could evaluate and tune such a system by having the user provide feedback on the quality q of each refinement, seeing for each of a set of values of m, and a set of values of n within each m, what the relation $q = F(m, n)$ is, empirically. A third parameter, k, could be the number of words that were put in the query to be *excluded* from the results.[12]

6.3.3 Top Correlates of Salient Words for "Jaguar"

Another approach is to see what words are correlated with any of the top words correlated with initial query word. For instance, the top 10 words correlated with the word "game" are: "jaguar," "played," "Atari," "the," "system," "developing," "boy," "console," and "control." (The word "boy" appears because the portable video game

[12]This idea for future work comes in part from Jude Shavlik.

"Game Boy" is on many Web pages that are about video games.) Clearly, other than "the" (which can be eliminated using a stop list), a query involving these words would pull out pages that are specifically about video games, and that mention the Atari Jaguar, while not pulling up pages from the other two major meanings evoked by the query "jaguar" alone. I have run such a query on Google and it does bring up a very focused set of pages.

Similarly, the top ten words correlated with the word "cat" are: "jaguar," "the," "box," "pictured," "black," "endangered," "pages," "animals," "coat," and "hat." This group of words does not make for a very good query for pulling out the original feline meaning of jaguar, because, it appears, too many of the automotive pages pull refer to cats as a metaphor. However, most of the pages that are pulled up by a Google query of "jaguar cat" are about the feline meaning.

The top ten words that are correlated with the word "type", which is the top-ranked word for the automotive meaning, are: "jaguar," "mk," "usa," "series," "xj," "sale," "the," "cars," "part," and "for." Doing a Google query with these words, minus "the" and "for" and plus "type" itself, leads to pages that are mainly about the car. Query refinement, of course, will work best if it uses a combination of automatic and user-driven selection of queries. A user who is a Jaguar automotive enthusiast will be able to select a good combination of these query words, even if she did not think of them initially. The system can prime her memory.

6.3.4 A HAL Matrix for "Ford"

Tables 6.13 through 6.16 show a single dimension, with respect to the word "Ford," of a HAL matrix constructed from a search engine result set of Web pages from a query on that word. As in the case of the "jaguar" pages, the "Ford" pages reflect an uneven distribution of keywords corresponding to the different meanings of the word "Ford." The largest number of keywords are related to the Ford Motor Co, Henry Ford, the Ford Foundation, and related topics. Then there are various other meanings, the most frequently reflected being that of President Gerald Ford, and then other figures such as the actor Harrison Ford, the director John Ford, the model Patricia Ford, and the author Ford Madox Ford. There is also Ford's Theatre, where Lincoln was shot. There are also Web pages about counties named Ford, which exist in Illinois and in Kansas, at least.

In terms of the two most common meanings of the word "ford" in terms of query results, the "Ford Motor" meaning is reflected first by word number two in Tables 6.13 through 6.16 , "motors," and the the "Gerald Ford" meaning is reflected best by the word "president," which is word number 19 in Tables 6.13 through 6.16. The word "library," which appears as word number ten in Tables 6.13 through 6.16, is not unambiguously about the Gerald Ford Presidential Library, because there are other libraries with "Ford" in their name that are named after other people named Ford.

6.3.5 Top Correlates of Salient Words for "Ford"

If the user identifies either of these two words—"motors" or "president"—for use in query refinement, the system can then suggest additional words. The top 15 words associated with "motors" are (after eliminating stop words): "company," "Ford," "racing," "firestone," "please," "affiliate," "iframe," "Ferguson," "models," "credits," "replacement," "club," "Henry," "vehicle," and "copyright." Most of these have an automotive meaning, and the user could select them for further query refinement. The top

Table 6.13: One Dimension of the HAL-Style Matrix Relative to the Word "Ford" (Words 1-50)

rank	word	score	*rank*	word	score
1	ford	11186	*26*	museums	996
2	motors	3688	*27*	bell	978
3	the	2932	*28*	all	974
4	part	2918	*29*	we	946
5	company	2816	*30*	informative	872
6	county	2727	*31*	owners	856
7	trucks	2574	*32*	vehicle	856
8	harrison	2081	*33*	capri	847
9	site	1910	*34*	credits	837
10	library	1783	*35*	exploring	822
11	foundation	1700	*36*	john	819
12	club	1528	*37*	home	817
13	web	1491	*38*	one	779
14	mercury	1424	*39*	film	778
15	pages	1407	*40*	movies	767
16	models	1404	*41*	made	762
17	cars	1307	*42*	mustang	752
18	click	1202	*43*	service	752
19	president	1162	*44*	pickups	744
20	fellow	1149	*45*	historical	730
21	news	1109	*46*	society	730
22	theatre	1108	*47*	racing	705
23	creek	1050	*48*	lincoln	703
24	tractor	1037	*49*	consul	692
25	taurus	1009	*50*	year	689

Table 6.14: One Dimension of the HAL-Style Matrix Relative to the Word "Ford" (Words 51-100)

rank	word	score	*rank*	word	score
51	patricia	683	*76*	price	562
52	pictured	676	*77*	books	552
53	working	673	*78*	related	547
54	engine	668	*79*	directions	546
55	this	664	*80*	fellowship	537
56	search	650	*81*	copyright	536
57	performance	650	*82*	globe	524
58	post	649	*83*	sale	523
59	school	646	*84*	register	523
60	colony	640	*85*	links	510
61	gerald	639	*86*	here	509
62	guests	639	*87*	no	503
63	inc	630	*88*	probe	501
64	fleet	612	*89*	celebrity	492
65	center	612	*90*	family	492
66	conservancy	611	*91*	about	490
67	diesel	605	*92*	com	486
68	america	586	*93*	for	484
69	biography	582	*94*	programs	483
70	fans	578	*95*	members	481
71	award	576	*96*	firestone	479
72	dealers	574	*97*	contact	458
74	rates	567	*99*	july	457
75	manager	564	*100*	accessories	456

Table 6.15: One Dimension of the HAL-Style Matrix Relative to the Word "Ford" (Words 101-150)

rank	word	score	*rank*	word	score
101	classic	451	*126*	galaxy	391
102	visiting	449	*127*	quote	390
103	born	448	*128*	henry	388
104	produced	445	*129*	power	385
105	mail	440	*130*	series	385
106	houses	438	*131*	you	384
107	ranch	437	*132*	list	380
108	gallery	432	*133*	sterling	377
109	events	432	*134*	tires	377
110	ranger	429	*135*	owned	376
111	councils	426	*136*	free	375
112	welcome	426	*137*	thanks	374
113	called	425	*138*	online	374
114	design	424	*139*	june	372
115	productive	423	*140*	filmography	371
116	replacement	423	*141*	co	371
117	photo	422	*142*	line	369
118	dissertation	419	*143*	custom	368
119	viewed	418	*144*	auto	367
120	if	411	*145*	automobile	366
121	south	408	*146*	support	365
122	mans	403	*147*	publish	363
123	he	402	*148*	madox	362
124	won	396	*149*	world	361
125	make	395	*150*	history	360

Table 6.16: One Dimension of the HAL-Style Matrix Relative to the Word "Ford" (Words 151-200)

rank	word	score	*rank*	word	score
151	campaigns	360	*176*	directors	317
152	rings	359	*177*	rights	316
153	ann	359	*178*	predoc	312
154	presidential	358	*179*	fact	311
155	webring	351	*180*	philes	304
156	and	351	*181*	joined	302
157	collection	351	*182*	internal	301
158	order	350	*183*	diss	300
159	day	349	*184*	party	299
160	laser	349	*185*	offers	299
161	places	341	*186*	administrator	297
162	email	339	*187*	slant	296
163	unit	335	*188*	garden	295
164	actor	335	*189*	policy	293
165	top	333	*190*	heritage	293
166	super	333	*191*	please	292
167	personally	332	*192*	thirty	290
168	comparing	328	*193*	congress	289
169	left	328	*194*	falcon	289
170	ltd	326	*195*	sables	288
171	association	325	*196*	ferguson	288
172	introduced	324	*197*	tech	287
173	canada	324	*198*	mason	287
174	automotive	321	*199*	locations	284
175	passenger	321	*200*	official	283

15 words associated with "president," again after eliminating stop words, are: "Ford," "Nixon," "presidential," "Gerald," "unit," "Lincoln," "national," "visiting," "american," "select," "history," "library," and "scouting." Most of these have a meaning associated with Gerald Ford.

6.3.6 A HAL Matrix for "Lincoln"

Tables 6.17 through 6.19 show the single dimension, with respect to the word "lincoln," of a HAL matrix constructed out of Web pages that were retrieved in response to that word. As in the case of the "jaguar" and "Ford" keyword sets, the "Lincoln" keywords do not reflect an even distribution of the various meanings of "Lincoln." Instead, they reflect the two most frequently occurring meanings, which are those reflecting Abraham Lincoln and Lincoln, Nebraska. Some of the keywords occur in several meanings; for instance, the word "county" appears in pages about Lincoln County, Oregon and also Lancaster County, Nebraska, which contains Lincoln, Nebraska. Keyword #4, "Abraham," pulls out the "Abraham Lincoln" meaning quite cleanly; keyword #30, "Nebraska," pulls out the "Lincoln, Nebraska" meaning, and keyword #54, "cars," pulls out the meaning that refers to the Ford Motor Company's Lincoln brand of cars. Other meanings occurring in the keywords in Tables 6.17 through 6.19 include the various other places named after Lincoln, various universities and colleges named Lincoln, and the Abraham Lincoln brigade, which fought in the Spanish Civil War.

Table 6.17: One Dimension of the HAL-Style Matrix Relative to the Word "Lincoln" (Words 1-50)

rank	word	score	*rank*	word	score
1	lincoln	19571	*26*	highway	1559
2	county	9965	*27*	continental	1529
3	the	7476	*28*	memory	1514
4	abraham	4333	*29*	online	1512
5	city	4036	*30*	nebraska	1503
6	center	3568	*31*	research	1484
7	president	3520	*32*	university	1473
8	site	2874	*33*	moved	1460
9	library	2714	*34*	houses	1435
10	assassin	2328	*35*	year	1362
11	national	2177	*36*	community	1316
12	life	1933	*37*	web	1311
13	election	1933	*38*	douglas	1306
14	news	1900	*39*	di	1254
15	club	1884	*40*	in	1248
16	school	1849	*41*	nes	1246
17	mark	1828	*42*	war	1185
18	illinois	1775	*43*	born	1164
19	home	1767	*44*	inaugural	1142
20	park	1761	*45*	address	1128
21	mary	1709	*46*	this	1102
22	family	1706	*47*	publicity	1078
23	informative	1690	*48*	gettysburg	1064
24	links	1655	*49*	live	1063
25	state	1626	*50*	all	1063

Table 6.18: One Dimension of the HAL-Style Matrix Relative to the Word “Lincoln” (Words 51-100)

rank	word	score	*rank*	word	score
51	speeches	1046	*76*	indiana	804
52	pages	1044	*77*	government	798
53	town	1030	*78*	quote	797
54	cars	1028	*79*	williams	788
55	todd	1025	*80*	board	787
56	student	1000	*81*	son	787
57	american	949	*82*	bytes	780
58	part	933	*83*	history	779
59	search	932	*84*	thomas	779
60	biography	918	*85*	historical	772
61	paper	892	*86*	law	770
62	logging	891	*87*	places	767
63	robert	874	*88*	our	766
64	regions	861	*89*	college	760
65	service	859	*90*	springfield	754
66	association	858	*91*	working	753
67	we	849	*92*	washington	747
68	locations	838	*93*	republican	743
69	civilization	825	*94*	questions	737
70	edition	821	*95*	john	729
71	offers	818	*96*	generally	724
72	unit	818	*97*	nominated	724
73	for	816	*98*	mans	705
74	one	816	*99*	campus	699
75	mr	807	*100*	websites	696

One Dimension of the HAL-Style Matrix Relative to the Word "Lincoln" (Words 101-150)

rank	word	score	*rank*	word	score
101	street	691	*126*	support	591
102	including	686	*127*	department	589
103	special	685	*128*	programs	587
104	tomb	680	*129*	virtual	585
105	day	678	*130*	father	576
106	books	675	*131*	and	575
107	mail	670	*132*	co	574
108	education	668	*133*	make	563
109	wife	664	*134*	union	558
110	vote	659	*135*	owners	555
111	rights	656	*136*	museums	552
112	debate	655	*137*	map	550
113	click	650	*138*	tad	546
114	phone	643	*139*	divides	544
115	commerce	639	*140*	brigade	540
116	study	637	*141*	save	534
117	medic	635	*142*	white	532
118	congress	627	*143*	death	529
119	chamber	625	*144*	slavery	525
120	resources	624	*145*	emancipation	525
121	married	616	*146*	business	521
122	beachey	615	*147*	times	521
123	gave	614	*148*	boyhood	520
124	photo	609	*149*	line	515
125	ford	602	*150*	april	512

Table 6.19: One Dimension of the HAL-Style Matrix Relative to the Word "Lincoln" (Words 151-200)

rank	word	score	*rank*	word	score
151	campaigns	360	*176*	directors	317
152	rings	359	*177*	rights	316
153	ann	359	*178*	predoc	312
154	presidential	358	*179*	fact	311
155	webring	351	*180*	philes	304
156	and	351	*181*	joined	302
157	collection	351	*182*	internal	301
158	order	350	*183*	diss	300
159	day	349	*184*	party	299
160	laser	349	*185*	offers	299
161	places	341	*186*	administrator	297
162	email	339	*187*	slant	296
163	unit	335	*188*	garden	295
164	actor	335	*189*	policy	293
165	top	333	*190*	heritage	293
166	super	333	*191*	please	292
167	personally	332	*192*	thirty	290
168	comparing	328	*193*	congress	289
169	left	328	*194*	falcon	289
170	ltd	326	*195*	sables	288
171	association	325	*196*	ferguson	288
172	introduced	324	*197*	tech	287
173	canada	324	*198*	mason	287
174	automotive	321	*199*	locations	284
175	passenger	321	*200*	official	283

6.3.7 Top Correlates of Salient Words for “Lincoln”

For the keyword “Abraham”, the top 15 correlated keywords, after removing stop words, are: “lincoln,” “president,” “mary,” “assassin,” “attending,” “born,” “national,” “online,” “research,” “school,” “year,” “life,” “todd,” “thomas,” and “paper.” Most of these are relevant to Abraham Lincoln. For the keyword “Nebraska”, the top 15 correlated keywords, after removing stop words, are: “lincoln,” “acting,” “forecast,” “southwest,” “iowa,” “city,” “local,” “game,” “interlinc,” “pm,” “cycle,” “clear,” “temperature,” “hixson,” and “sports.” These words appear because many of the pages are about weather or sports (in Nebraska), and because Lincoln, Nebraska is near Iowa. For the keyword “cars”, the top 15 correlated keywords, after removing stop words, are: “lincoln,” “club,” “cartier,” “year,” “mark,” “mercury,” “special,” “resources,” “added,” “show,” “continental,” “edition,” “marketing,” “inventories,” and “safety.” For the most part, one can see how these words are related to the automotive meaning of “Lincoln.” Some of them are clear upon further investigation; for instance, “Cartier” refers to a special edition of the Lincoln Town Car.

6.3.8 *HAL-Set-Partition*: An Algorithm for Finding Sets of Semantically-Related Words

A simple algorithm, which I will call *HAL-Set-Partition*, for separating out the sets of words associated with each meaning of the query is the following. First, eliminate all of the words on a stop list from the set of words under consideration. Then consider any two words as “connected” if one of them appears within the top n words correlated with the other, where n is a tuning parameter, disregarding self-links and links to the

Figure 6.11: *HAL-Set-Partition:* An Algorithm for Finding Sets of Semantically-Related Words

1. Consider a Web query on a single word, such as "jaguar".
2. Consider a set of m Web pages that are the result of such a query.
3. Constuct a HAL matrix out of the words on these Web pages. (See Section 2.9.2 for a description of how this is done).
4. For each word w in this matrix, consider the top n other words w_j in the matrix correlated with it, in terms of the values in w's dimension of the HAL matrix. Consider w and each w_j to be connected within an undirected graph.
5. Consider each of the connected components in this graph to represent a set of semantically-related words.

original query word (here, "jaguar"), since the original query word is likely to connect all these sets. Then construct sets out of the connected components of the graph that is implied by these connections. This algorithm is analogous to the one we have discussed above (described in Figure 6.5) that uses the link structure of the pages to construct related sets. *HAL-Set-Partition* is given in pseudocode in Figure 6.11.

Figure 6.12: A Hypothetical Graph Output of *HAL-Set-Partition*

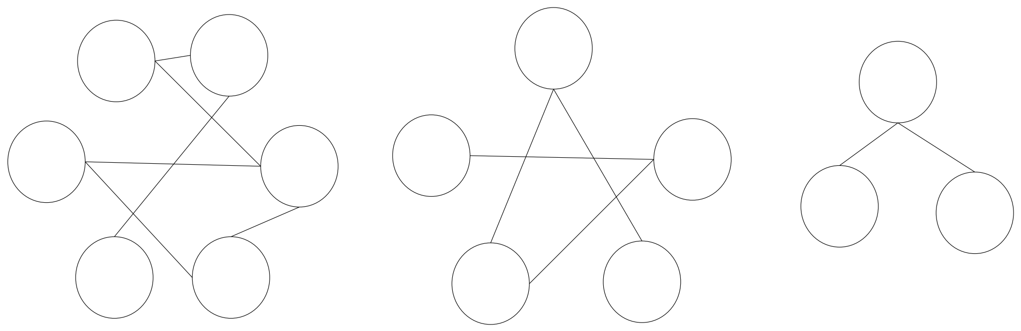


Figure 6.12 shows a hypothetical graph output of HAL-Set-Partition. In this figure, the circles represent distinct words. The connections represent pairs of words for which one of the words appears on the list of the top n HAL-matrix correlates of the other word.

A problem is that if that there are connections between the sets, such as that which would be caused by the metaphorical use of the word "cat" in the automotive context, then this algorithm may not work well. Perhaps, the use of the tuning parameter may be able to compensate for this, by limiting the number of connections, but this may also have the effect of disconnecting components that should be connected.

6.3.9 Application of *HAL-Set-Partition* to the "Jaguar" Keywords

HAL-Set-Partition does not work particularly well for $n = 2$ or $n = 3$ in the case of "jaguar," although it works better for $n = 2$.

For $n = 3$, it separates all the words (341 of them that are among the top three linked from the top 200 linked to "jaguar" itself) into 30 sets, which is many more than would be desired. All of these except one have between two and five members, and most of these small sets are not particularly good in terms of giving a a distinct feel for the topic. There is also a large set with 251 members, and this contains words from the animal, automotive, and video game senses of the word "jaguar," as well as at least one other sense, a Fender electric guitar named Jaguar.

For $n = 2$, there are 271 words, drawn from the top 2 linked from the top 200 linked to "jaguar" itself. Here, the algorithm does somewhat better. There are even

Table 6.20: *HAL-Set-Partition* Results for “Jaguar”

n	number of words	number of sets	set sizes
2	341	30	2-5 (29 sets); 241 (1 set)
3	271	86	2-5 (84 sets); 46 (1 set); 78 (1 set)

more sets, 86 in fact, which is what you would expect because the graph is not as well-connected. As before, the vast majority of these—all except two sets—have two to five members and do not generally do a good job of evoking any of the main senses of “jaguar,” although the words within each set are typically related to one another. However, there are two sets that do a good job of evoking two of the senses of the word “jaguar,” the automotive sense and the video game sense. The set that evokes the automotive sense has 46 members, and virtually all of these would be found on Jaguar automobile page; one word that would not is “guitar,” which appears because it is linked to the automotive set through the word “body,” since both guitars and cars have bodies (and because there was a Fender electric guitar named the Jaguar). The set that evokes the video game sense has 78 members, and mainly contains words that would correspond to this sense, although there a few words, like “showroom” and “service” that probably should have been associated with the automotive meaning. “Showroom” is linked to “service” (since car dealerships have service areas and showrooms), which is linked to “network,” which is linked to “game.” Jaguar automobiles have a service network, and video games can be networked, so “network” should really appear in both sets (with two different meanings), but this algorithm does not allow it.

The reason why the original, animal, sense of “jaguar” is split between so many small sets is because there are relatively few words among the top 200 words in the jaguar HAL list that relate to the animal meaning, compared to the other two meanings.

6.3.10 Application of ***HAL-Set-Partition*** to the "Ford" Keywords

I tested the *HAL-Set-Partition* algorithm with the "Ford" keywords, again for $n = 2$ and $n = 3$. In this case, the algorithm performed worse than it did in the case of the "jaguar" keywords. For the case of $n = 2$, there was only one large set, with 142 members out of the 294 possible, but this set contained keywords from the various meanings of "Ford." All of the other sets were relatively small, and few of them appeared that they would function well in teasing out any particular meaning of "Ford." However, there were these few: one of these was the set containing the three keywords "ranger," "aerostar," and "econoline," which are are all Ford motor vehicle models.

For the case of $n = 3$, again there was just one large set, with 270 out of 381 possible keywords, but again this large set combined several meanings of "Ford". Again, all the other sets were much smaller, and most did not evoke any particular meaning of Ford very well. One that did was a set that contained the keywords "diesel," "power," "turboramair," "turbo," "strokes," "supply," "western," "history," and "american," most of which are words that are heavily associated with a particular kind of Ford diesel engine. This is an example of how some topics are hidden within a query and can be identified with a method such as the one that we are discussing here.

6.3.11 Application of *HAL-Set-Partition* to "Lincoln"

As in the case of "jaguar" and "ford", the *HAL-Set-Partition* algorithm did not work particularly well. For $n = 2$, there was a single large set and many small sets. The large set had 100 items (out of 268 possible) and the vast majority of them were relevant to Abraham Lincoln, but there also were a few items about cars and other miscellaneous items. Few of the other sets evoked any of the other meanings of Lincoln well. For $n = 3$, there was again one large set and quite a few small sets. The large set contained 272 of the 335 possible members, and was dominated by the Abraham Lincoln meaning. Again, few of the other sets evoked any of the other meanings of Lincoln well.

6.3.12 *HAL-Disjoint-Supervised*: A User-Supervised Approach to Creating Lists of Words for Use in Query Refinement

Another approach to separating these word lists into semantically-related groups is the algorithm I call *HAL-Disjoint-Supervised.* This approach uses user supervision to create disjoint sets. That is, after you find the top correlates for each of the user-selected top words for each meaning (here, I selected, as I did above, "game," "cat," and "type," for each of the three meanings for "jaguar" respectively, although any other three words that elicit the three meanings would work just as well), the computer eliminates from each list of correlates any terms that are on the stop list and any terms that appear on more than one list. See Figure 6.13 for a pseudocode

Figure 6.13: Algorithm *HAL-Disjoint-Supervised*

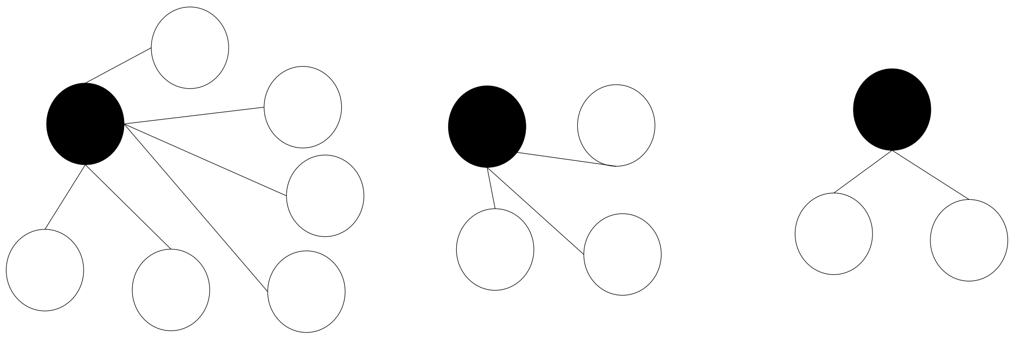
1. Compute the dimension of HAL matrix with respect to a word w and a set of Web pages on which that word appears (typically a set of Web pages that are the result of a web query on w). That, the HAL matrix is computed from the set of documents, and the dimension of the matrix relative to w is considered.
2. Have the user manually select a set of words from among the top j correlates (where j is a tuning parameter) of w that reflect all the meanings of interest. Call this set M.
3. For each of the words m in M, find its correlates. Call this set $C(m)$.
4. For each of the words m in M, let $D(m)$ be the set of words in $C(m)$ that are not correlates of any of the other words in M. The $D(m)$ can then be passed back to a search engine to refine the query.

description of this technique.

Figure 6.14 shows some hypothetical graphs that are output from *HAL-Disjoint-Supervised*. In this figure, the circles represent words, and the black circles represent user-selected words. The connections are present when a user-selected word w has another word (represented by a white circle) that is among the top j correlates of w but is not among the top j correlates of any other user-selected word.

HAL-Disjoint-Supervised does, in my opinion, an excellent job of separating out the different meanings of the term jaguar, albeit at the cost of a little user supervision. It leaves to the user what she does best; selecting terms related to each distinct meaning, and it leaves to the machine what it does best; computing the HAL term correlates and creating the disjoint sets. The only cost is that terms that might legitimately be left in more than one set are eliminated, but of course this may actually be a blessing, because if the terms are then passed back to the search engine

Figure 6.14: Hypothetical Output of *HAL-Disjoint-Supervised*



to get a more focused set of pages, the terms that are on more than one list are likely to cause trouble if they are included.

6.3.13 Application of *HAL-Disjoint-Supervised* to the "Jaguar" Keywords

Acting as the supervising user, I selected "game," "type," and "cat" as the keywords for use by HAL-Disjoint-Supervised, as words that I felt would pull out three distinct meanings of "jaguar."

The top ten words correlated with "game," after applying *HAL-Disjoint-Supervised*, are "boy," "gear," "interactive," "times," "overlays," "Hasbro," "player," "paused," "publish," and "headquarters." The word "boy" appears, as noted above, because of the fact that many of the game pages mention the "Game Boy," another video game. "Game gear" is a kind of video game. "Times" appears because there is a publication called the "Atari Times." "Overlays" appears because of overlays that are placed on the keyboard to play different games. Hasbro appears because it acquired Atari.

"Headquarters" appears because people often title their pages "game headquarters" or some such.

The top ten words correlated with "type", after applying *HAL-Disjoint-Supervised*, are "mk," "xj," "xke," "dinky," "xk," "roadster," "models," "luxury," "coupe," and "daimler". The abbreviations stand for different Jaguar automobile models. Most of the other words' automotive associations are obvious. "Daimler" was a car model associated with Jaguar. Dinky is a manufacturer of models of Jaguar cars.

The top ten words associated with "cat", after applying *HAL-Disjoint-Supervised*, are "endangered," "coat," "hat," "leopards," "big," "horace," "wild," "species," "bird," and "western." Several of these words, such as "endangered" and "leopards," have an obvious association with the jaguar (cat). "Coat" appears because of the jaguar's coat; "big" appears because the jaguar is a big cat. "Hat" appears because there are various pages about cats referring both to the "Cat in the Hat" (by Dr. Seuss) and jaguars, among other pages that also have both words. It is not clear why "horace" appears. "Western" appears because jaguars are native to the Western Hemisphere. "Bird" appears because many of the pages that describe jaguars also describe various other animals, such as birds.

Thus we can see that *HAL-Disjoint-Supervised* does an excellent job, at least with this example and this supervision. It might make a good substrate for guided query refinement, allowing users to use the facilities of the machine to identify related topics and to better focus queries.

HAL-Disjoint-Supervised also considerably reduces the number of words that are considered correlated with any central word. In the case of "game", there were 1467 words correlated with it before the formation of the disjoint sets, and 973 (about

Table 6.21: Reduction in Size of Keyword Sets for Various Meanings of “Jaguar” after Making the Sets Disjoint

central keyword	non-disjoint size	disjoint size	percent of original size
game	1467	973	66%
type	903	525	58%
cat	356	169	47%

66%) of these were left after the intersecting words were thrown out. For “type,” these numbers were 903, 525, and 58%; for “cat,” they were 356, 169, and 47%. See Table 6.21 for a summary of these statistics. The relative magnitude of these terms also gives one something of a sense of the relative prevalence of each of the meanings in the underlying sample of Web pages, and to a lesser extent, to the universe of pages containing “jaguar” that lies behind it.

HAL-Disjoint-Supervised also suggests an alternative method for the ranking of pages relative to a particular topic. Once a particular disjoint set of keywords is determined that describes that topic, then each candidate page is given a score which is a weighted sum of which of these keywords appear on that page, with a higher weight being given to those keywords that are better correlated with the central keyword. The pages are then ranked according to their score. You would need to correct for the length of the page, so that lengthy pages did not get higher scores. This technique would have the advantage of being less brittle than conventional search engines, which typically *require* that *all* the terms in the search query be in the result pages (although some allow disjunctive queries as well).

Another way to rank pages that would be even less brittle would be to take account of the conditional probabilities that individual keywords would appear in pages in a particular class, not requiring them to be assigned to one class or another (as is required by an algorithm based on disjoint sets).

6.3.14 Application of *HAL-Disjoint-Supervised* to the "Ford" Keywords

I tested the *HAL-Disjoint-Supervised* algorithm for the Ford data set using the central keywords "motors," "president," and "Harrison." (This last keyword refers to the actor Harrison Ford.)

For the keyword "motors", the top ten words after eliminating the intersection with the other sets were: "racing," "Firestone," "affiliate," "iframe," "ferguson," "models," "credits," "replacement," "Henry," and "club." For "company," "racing," "Firestone," "henry," and "models," the connection with the Ford Motor Company is obvious. "Iframe" appears to have missed the stop list of HTML words. "Affiliate" appears because automobile companies often affiliate with other automobile companies (Ford has a relationship with Mazda). "Credits" may appear because of environmental tax credits associated with certain "green" vehicles, and because of its singular form, "credit," since many pages are about getting credit to buy a vehicle. "Replacement" appears because of the description of replacement parts on some pages. "Club" appears because of various clubs formed out of interest in Ford cars. So, as we see, most of these terms are about the automotive meaning, and none are about the other two meanings in question.

For the keyword "president", the top ten words after eliminating the intersection with the other sets were: "Nixon," "presidential," "Gerald," "library," "scouting,", "rockefeller," "houses," "speeches," "Omaha," and "archives." For most of these, the connection with President Gerald Ford is clear. The words "library" and "archives" appear because there are libraries and archives with information about President Ford. "Scouting" appears because Ford had been an Eagle Scout. "Rockefeller" appears because Nelson Rockefeller was Ford's vice-president. "Houses" appears because President Ford spent most of his political career in the U.S. House (plural and singular versions of words only appear as a single token in the list). Omaha is Ford's birthplace.

Thus, *HAL-Disjoint-Supervised* does a good job on separating out keywords relevant to Gerald Ford.

For the keyword "harrison", the top ten words in the disjoint set are: "gallery," "imdb," "pictured," "filmography," "tv," "info," "Ann," "worshiped," "dvd," and "entertainment." For most of these, it is easy to see why these terms would be more likely to appear on pages about Harrison Ford rather than the other 2 meanings considered here. The word "info" appears to occur by random variation in the sample. "Ann" may be a reference to Anne Heche, Ford's co-star in one movie, or the middle name of his biographer. "Imdb" is a reference to the Internet Movie Database, a popular site that lists information about movies and movie stars.

So, *HAL-Disjoint-Supervised* does a moderately good job of pulling out words that are relevant only to Harrison Ford in comparison to the other two meanings, although these keywords would not be very specific to him if we were trying to distinguish him from other movie stars.

Again, *HAL-Disjoint-Supervised* reduces the number of keywords associated with each set substantially. For the keywords centered on "motors," the number of cor-

Table 6.22: Reduction in Size of Keyword Sets for Various Meanings of “Ford” after Making them Disjoint

central keyword	non-disjoint size	disjoint size	percent of original size
motors	313	183	58%
president	645	435	67%
Harrison	514	319	62%

related keywords was reduced from 313 to 183, or about 58 percent of the original value. For “president,” these numbers were 645, 435, and 67 percent. For “harrison,” they were 514, 319, and 62 percent. These data are summarized in Table 6.22.

6.3.15 Application of HAL-Disjoint-Supervised to “Lincoln”

I tested the *HAL-Disjoint-Supervised* algorithm on the keywords “Abraham”, “Nebraska”, and “Cars”, as above. For the keyword “Abraham”, the top ten correlated keywords in the disjoint set are: “mary,” “assassin,” “attending,” “born,” “national,” “school,” “todd,” “thomas,” “paper,” and “historical.” This is quite similar to the set listed above for the non-disjoint set, indicating that for this meaning of “Lincoln” at least, the sets were well-separated to begin with. It is easy to see how most of these terms are related to Abraham Lincoln. “Mary,” “Todd,” and “Thomas” were names of people in Lincoln’s family.

For the keyword “Nebraska”, the top ten keywords were: “acting,” “forecast,” “southwest,” “iowa,” “local,” “game,” “interlinc,” “pm,” “cycle,” and “clear.” Many of these are weather-related terms, since the search happened to bring up a lot of pages about weather in and around Lincoln, Nebraska. “Acting” appears because of a theater school at the University of Nebraska. “Iowa” appears because Lincoln,

Table 6.23: Reduction in Size of Keyword Sets for Various Meanings of "Lincoln" after Making them Disjoint

central keyword	non-disjoint size	disjoint size	percent of original size
Abraham	946	734	78%
Nebraska	358	217	61%
cars	259	153	59%

Nebraska adjoins Iowa.

For the keyword "Cars", the top ten keywords were: "club," "cartier," "mercury," "special," "added," "show," "continental," "inventories," "webring," and "safety". Apart from "webring," which appears because one of the Lincoln cars' pages is linked into a webring, the automotive meaning of most of these words is clear.

In this case, requiring that the sets be disjoint does not seem to add much to the semantic separation, which is good even without the disjoint sets.

Again, the size of the lists of keywords was reduced as a result of the separation into disjoint sets. For the keyword "Abraham," there were 946 correlated keywords in the non-disjoint set, and 734 in the disjoint one, about 78 percent of the original size. For "Nebraska," these numbers were 358, 217, and 61 percent. For "cars," they were 259, 153, and 59 percent. These data are summarized in Table 6.23.

Thus, we have seen that the use of a method based on semantic networks, user supervision, and disjoint keyword sets drawn from these semantic networks, can be effective in separating meanings from ambiguous web queries. The words found in these sets can be used to refine the queries and get more precise results.

6.4 Review of the Contributions of this Chapter

In the first section of this chapter, I showed that, given a set of Web pages that are the result of a search engine, separating this set into connected subsets can be an effective technique for semantic separation of unrelated topics. I discovered that simply doing, this, however, is not enough; pages are often connected via highly-referenced pages. Thus, a contribution of this section is the discovery that, by pruning out connections to these pages, performance on semantic separation is improved.

In the second section of this chapter, I experimented with the use of semantic neighborhoods in WordNet to identify different meanings in ambiguous search engine results. The contribution of this section is the formulation of an effective algorithm to do such separation, which was tested with two sets of results. This algorithm is based on creating a vector representing the semantic neighborhood of each meaning of a word, and then using the dot product with this vector to test each page for similarity with that meaning; the meaning with the highest similarity is returned.

In the last section of this chapter, I experimented with using a HAL-based matrix of correlations between keywords to identify sets of keywords that could be used for query refinement. Such refinement could be used to remove ambiguity from search engine results. The main contribution of this section is the finding that a semi-automated method, whereby the user selects one keyword characteristic of each meaning of interest, is best in identifying a set of Web pages corresponding to each meaning. Some of the keywords in each set could then be passed to a search engine as a refined query in order to obtain a less-ambiguous result.

Chapter 7

Emerging Forms of Collaboration and Filtering

7.1 Introduction

Up to this point, we have considered mechanisms for better organization of information on the Web, using such information as manual page classifications into hierarchies, textual information on the pages themselves, and user ratings of Web pages. However, we have not considered more explicitly collaborative systems, in which both humans and information items (e.g. Web pages) are explictly represented as part of the system. I review some of the systems that have already been built along these lines in Section 7.2. In Section 7.3, I discuss a linear-relaxation-based system that I have built that could potentially be used to organize communities consisting of information items, their authors, and ratings of information items by people in the system other than their authors. This system has the potential to democratically filter information items so as reduce the flood of information, and also makes document author reputations (“karmas”) explicit.

7.2 Current Collaborative Systems

Since the construction of the GroupLens system for the collaborative filtering of Usenet articles, a number of independent Web sites have popped up that feature some combination of collaboration and filtering. Three of the best known of these are the non-profit sites Advogato (www.advogato.org), Slashdot (www.slashdot.org), and Kuro5shin (www.kuro5shin.org). Epinions (www.epinions.com) and Plastic (www.plastic.com) are commercial sites sharing some of the characteristics of the non-profit sites. My suspicion is that these systems emerged in part due to the dissatisfaction of many people with the low signal to noise ratio that had developed within Usenet News after the Internet became widely popular; Usenet became inundated with low-quality messages and commercial spam. Many of the developers of the new conferencing systems also made their software available as open source.

It may be possible to move away from a centralized system of service delivery. Singh et al. [114] discuss the theory of how individuals working in communities can provide useful services to one another and validate trust. (Here, services are anything from advice on what restaurant to choose to actual physical services, such as plumbing or piano teaching.) There are many books that discuss how to build online communities from a more practical perspective. Two such books are by Greenspun [45] and by Kim [57].

Filtering of information from the Web can occur on both the individual and the group level. At the individual level, there has been a good deal of work on the development of agents that can take feedback from users and customize information delivery for each user's needs. Some examples include the work of Shavlik et al. [112], Goecks and Shavlik [40] and Edwards et al. [33].

Terveen and Hill [119] report their experience with Phoaks (People Helping One Another Know Stuff), a system that they developed which mines Usenet postings for URLs and uses these as recommendations to the community of Phoaks users. They define collaboration as occurring when sites cross-reference one another. They find that the more commercialism there is, the less collaboration there is. In terms of the Web graph, this means that in non-commercial communities, such as communities of interest such as "Beat Poets," the graph represents a relatively tightly connected component. Alternately, in a commercial area such a "Italian travel" (where there are many promotional sites) there are are a lot of sites that sit all off by themselves, and are not connected to any others. Italian travel probably has both commercial and non-commercial sites, and the non-commercial ones are more likely to link to one another. Terveen and Hill find a relationship between highly connected sites and assessments of site quality. This is an interesting situation; competition and commercialization are supposed to lead to quality, but here is a situation where information sharing seems to trump any advantage of competition.

All of these sites use ideas from collaborative filtering, the web of trust, or both. The web of trust is a concept borrowed from public key cryptography [18]. Instead of the trust relation being one of authenticating that someone is who they say they are, however, it now becomes that you trust them in general, that is, you trust their opinions, you trust them not to lead you astray. Obviously, this is a more fuzzy concept than the concept in cryptography—after all, someone is either who they say they are, or they are not, but there are degrees to which you trust different people or institutions. However, both imply a certain level of transitivity to the relation; if A trusts B, and B trusts C, then A is more likely to trust C. However, the relation is typically not reflexive; if A trusts B, B does not necessarily trust A. In fact, insofar as societies are organized into hierarchies, often the trust relation is not reflexive, since less knowledgeable people lower down in the hierarchy (e.g. students) will be less trusted than those higher up in the hierarchy (e.g. teachers). However, the Internet presents an opportunity for those who have accumulated knowledge by hook or by crook to possibly trump those associated with more formal institutions. Zuboff [132] has argued that this can happen within a wide variety of organizations.

Slashdot is probably the best-known collaborative filtering site among the technical community, especially among open source advocates. It has tweaked its design over time. Gladwell [38] notes that small differences often make a huge difference in the popularity (and thus, implied usefulness) of a design. One example he gives is of the television show Sesame Street, which was not a success when it was first introduced. The show's producers found that kids stopped watching when people were on the screen talking, but the kids resumed watching when the Muppets were introduced into the group. Thus, small differences in the design of a collaborative filtering system may make a big difference in its success.

Slashdot evolved its web of trust over time. Slashdot uses the concept of "moderation points." Individuals who use the system are granted a certain number of moderation points periodically, based on such factors as how long they have been members of the system and their "karma," which is a measure of the number of moderation points their comments have been given by other users of the system. Moderation points are handed out sparingly and can be used to boost or downgrade the ratings of articles and comments written by others. Since moderation points are treated by the system as a scarce resource, and have some status associated with their allocation, Slashdot seems to have managed to reverse the usual attitude of users toward rating items in a system. Generally speaking, the "dirty little secret" of collaborative filtering systems is that the ratings matrix is quite sparse; people are not willing to spend a lot of time rating other people's postings, or other items that the system maintainers want to be rated. Slashdot seems to have gotten around this by two mechanisms: by creating a sense of community around membership in the system, and making the moderation points a scarce commodity that are associated with one's status (karma) in the system.

The history of the evolution of Slashdot's web of trust is interesting. They started with a small, 25 member, group of trusted individuals who were the only ones given any moderation points. The karma of all users in the system was initially based on the allocation of moderation points by this core group. As the system grew, it became clear that the original group of moderators could not keep pace with the number of articles posted, so the moderation group was expanded to include 400 members who had accumulated relatively high karma. Some of these people turned out to be bad apples that needed to be banned by the core group, but for the most part, it worked well. In the final stage, all users of the system were occasionally granted moderation points, based on how long they had been members of the system, and their karma. The system is so large now that it is difficult to attack the group with high karma; one would need an organized conspiracy of a large number of malicious users to do so. Some users have tried to set up a large number of bogus accounts to attack the system, but this has been detected because it was done in a unintelligent manner.

Slashdot still relies on a core group of editors (those associated with the group that started the system) to select the main topics for discussion, so it is still somewhat elitist in that sense. Any one can propose a topic for discussion, though. Kuro5hin, which is a system which is also quite popular, has some differences from Slashdot. Instead of having a panel of editors select the articles, users submit articles to a moderation queue under one of several topics. Other users moderate articles in these queues, as the top-rated articles emerge from the queue to appear in the appropriate category of the site. Also, Kuro5hin has a broader charter than Slashdot. Slashdot is devoted almost exclusively to technology, while Kuro5hin discusses technology, culture, politics, and other topics.

Kuru5hin uses the concept of "mojo" instead of "karma." The karma of a Slashdot user is the sum of the moderation points that other users have given to her postings. In Kuro5hin, mojo is a time-decaying average of the mean ratings of a user's postings. Thus mojo measures "what have you done for me lately?" as opposed to karma, which is more cumulative.

All of these systems consist primarily of posted information, individuals, and various relations between and properties of these. One can generalize this to the following: all postings, computer files (such as word processing documents, Acrobat files, mp3 files, etc.), or Web pages (chunks of information) can be generally be thought of as information items, members of a set called I. People can be thought of as members of the set of all people P, and institutions or groups of people, are members of the power set of P, which we can call G. Individuals, people, and groups are simply subscripted members of each of these sets.

Then there are several important relations that exist between these entities. One important relationship is *authorship*, which is typically made explicit within systems like Slashdot, Kuro5hin and Usenet (since users are authors of messages on these systems, and what is being accessed is the quality of these messages) but is usually not made explicit (at least not at a level that can easily be understood by the machine) with Web pages, even though it is always the case that a Web page is authored by an individual or group. Another important relationship is *readership*, which is also not explicitly made explicit by the Web, because even if the Web browser software knows the name of the user using it, it typically does not disclose this information to the Web server (which typically only logs the browser type, page retrieved, time stamp, and IP address of users downloading pages) for privacy reasons. Optionally associated with each readership relation can be an *evaluation* (rating), in which an individual associates a rating with an information item. This is done in a system such as Slashdot, but not typically done on the Web in general.

Another relation that can exist is a direct *trust* relationship between individuals, such as exists between friends or colleagues. There can also be a *referral* relation where individual A refers information item X to individual B. There can be various relations between information items themselves; for instance, one information item could be a *part* (subset) of another, or one can *link* (via http) to another, or one can be a *response* to a another in a threaded discussion. Finally, an information item can be *typed*, that is, it can be placed in a set, or in a hierarchy, in the manner that Yahoo! or Dmoz's editors do. Most information on the Web has not been typed, as many people, such as Pirolli et al. [90] have pointed out, and it is a losing battle to keep up using a relatively small group of people (that is, small relative to the number of Web pages). There is no reason to leave this to elites; the process of classification can be left to the end user, as is the process of evaluation, with the appropriate weighting by trust. Essentially, Dmoz and Yahoo! use self-selected gate-keepers to establish trust; this can be established in a more decentralized manner, as Kuro5hin has shown. XML [12] may provide a capacity for the authors of pages to classify them, but I doubt all authors will do so. XML represents a technical mechanism that can implement any bureaucratic mechanism, but the Web is lacking in uniform bureaucratic mechanisms, except that that is implicit through browser design.

In the next generation of the Web, there may be an attempt to deal, at least, with the problem of authorship. Authorship may be established through a combination of improved user authentication, and XML Web pages that explicitly contain a field for an individual or organizational author. Readership and evaluation is more difficult to solve, because people are less willing to give up their privacy, even if they gain something by participating in collaborative filtering. Here, though, systems like Slashdot have shown the way, and can be extended to the entire Web, if the social barriers to doing so can be overcome. The increasing popularity of directory management tools such as LDAP (Lightweight Directory Access Protocol [52]) allow many of the entities, such as organizations, people, computers, files, and Web pages/resources to be listed in directories and relations between these also be listed. This may facilitate the transition. While LDAP was devised to serve a directory of users, groups, and resources, authenticate users, and control user access to resources, the specification is quite general and could be adapted to the sorts of purposes I am discussing here. For instance, each (authenticated) individual could have an LDAP record with pointers to all of the Web pages she is the author of. Users could surf the Web one of two ways, anonymously, or authenticated. If they surf anonymously, ratings of a particular Web page are only accepted once from each network address (IP).

I argue that much of what has been written about search engines, collaborative filtering, digital libraries, distance education, and trust in communication networks can actually be generalized in terms of the framework and relations given above. In fact, any situation in which people are communicating and working together with information– which is most human situations today– can be improved via collaborative filtering using an *authorship-readership-rating* system, which is a simple extension of the Web as it exists today.

7.3 A Model Collaborative System

I have built a simple model that embodies some of the relations given above, and it is described in this section. The purpose of constructing it is to show the feasability of a collaborative system that supplements the existing Web with ratings of Web pages and of their authors. Such a system would be a useful addition to the existing methods of filtering pages on the Web through Web directories and search engines. A system based on this model could be used for a collaborative filtering system like Slashdot, or it could be a applied to the Web as a whole, if the authorship of pages could be determined in a standard manner.

For a set of people S rating information item k authored by person p, where p has authored a set of information items A_p (including k), and where the rating made by individual i of information item k is r_{ik}, and the overall rating of each article is

Figure 7.1: An Algorithm for a System of User and Document Ratings

1. Collect all the user ratings r_{ik} of all the documents k in the system where i is the user rating document k.
2. Set the initial overall rating of each document k, $rating_k$, using Formula 7.1.
3. For each user p in the system, set $karma_p$using Formula 7.2, based on all of the documents she has authored.

$rating_k$, and the karma of each individual p is $karma_p$, let

$$rating_k = \frac{\sum_{p \in S}(karma_p r_{ik})}{\sum_{p \in S} karma_p} \tag{7.1}$$

and let

$$karma_p = \frac{\sum_{k \in A_p}(rating_k)}{|A_p|} \tag{7.2}$$

Note that these formulas depend on each other. We can also, without loss of generality, assume that all values are positive (users can represent low karma and ratings with low positive values rather than negative values). The r_{ik} are given by the users. If we allow the users to start out with everyone having equal karma, before the first iteration of formulas 7.1 and 7.2, then after one iteration an initial value of the overall rating of each information item and of each individual's karma has been set. This technique is given in pseudocode in Figure 7.1.

Graphically, such a system could be visualized as a directed graph or network consisting of two types of nodes, representing individuals and documents, and two types of directed links (e.g. the two-individual, six-document system shown in Figure 7.3 below). One type of link represents the authorship relation, and points from a document to its author. The other type of link represents the act of rating a document, and points from an individual to a document. This latter link is labeled with the numeric value of the rating. Thus this network is somewhat more complex than the current Web, although not as complex as proposals such as the semantic Web [10] (in which semantic information is attached to every Web page), into which such a network could be embedded.

After one iteration of Equations 7.1 and 7.2, we could stop; but we have only captured first-order effects. We have not allowed information to propagate in the network more than one step. After one step, we have done what many search engines other than Google do; they count the in-links coming into a page. (This has been slightly modified, since it is a weighted count, based on the ratings.) Google's designers make a good argument that second, third, and nth-order information is needed in the simple in-link count context; the context described above is only slightly more complex, because it is still linear—instead of a simple sum of in-links, it is a weighted sum of ratings. But the need still exists to propagate data forward through the network more than one step, and an iterative algorithm, similar to that made by Google, can do this.

One way to make the information propagate in the system would be by using a pair of linear recurrence relations, iterated until they converge (given below in Formulas 7.3 and 7.4). In these relations, the rating of a document is a function of all of the ratings of people that are rating it, weighted by those individuals' karmas (and normalized). The karma of an individual is then the mean of the ratings that that person authored. Initially, the document ratings are unweighted, and all the individuals have equal karma. Then, the document ratings are re-computed based on the weighted karma of the rating individuals, and then the authors' karma is re-computed, iteratively. Unrated documents are set to an average karma on all iterations (in my experiments, five on a scale from one to ten). Since the system is linear and the equations are normalized, it can be expected to converge, and the result is a relaxation-type algorithm akin to some algorithms used in computer vision[1], unlike non-linear systems which may exhibit chaotic behavior [39].

Formulas 7.1 and 7.2 are used to initialize the system. The two formulas are now modified so that the document ratings are weighted and the subscripts are expanded to reflect the iterations, thus transforming the formulas into two linear recurrence relations:

$$rating_{k(m+1)} = \frac{\sum_{p \in S}(karma_{pm} r_{ik})}{\sum_{p \in S} karma_{pm}} \tag{7.3}$$

$$karma_{p(m+1)} = \frac{\sum_{k \in A_p}(rating_{km})}{|A_p|} \tag{7.4}$$

A pseudocode version of this algorithm is given in Figure 7.2.

I did a simple experiment with a three-person system to illustrate the effects of democracy in such a system. In this system, there were three people, each was the author of two documents (for a total of six), and each rated all four documents that she was not the author of. Person 1 rated all four documents written by the other two people with a rating of 1 on a scale from 1 to 10. Persons 2 and 3 both rated both of the documents that person 1 wrote with a rating of 1. Person 2 and 3 rated each other's documents with a rating of 10. Thus persons 2 and 3 represent a "mutual

[1]Of course, linear recurrence relations need not converge; for example, the equation $x_{i+1} = 2x_i$ diverges to infinity. But since these equations are normalized, they do not diverge, and because they are linear, they are not chaotic. It would not be a good idea to use a nonlinear recurrence relation for constructing a relaxation-based collaboration system, unless you could prove that its behavior would not be chaotic.

Figure 7.2: Relaxation Algorithm for a System of User and Document Ratings

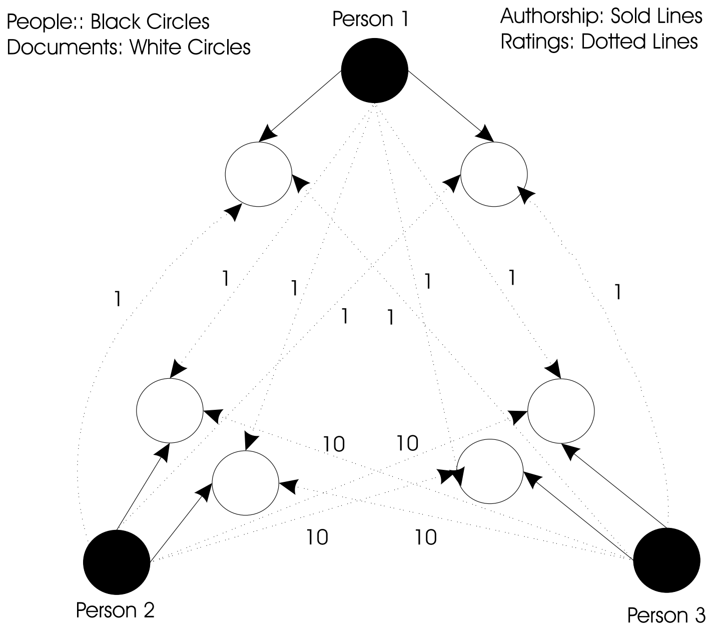
1. Collect all the user ratings r_{ik} of all the documents k in the system where i is the user rating document k.
2. Set the initial overall rating of each document k, $rating_k$, using Formula 7.1.
3. For each user p in the system, set $karma_p$using Formula 7.2, based on all of the documents she has authored.
4. Repeat, with iteration counter m, until convergence (that is, until values of $karma_{pm}$ and $rating_{km}$ do not change more than a small constant ε). Use Formulas 7.3 and 7.4 to update these values.

admiration society." Person 1 thinks poorly of their documents, and they think poorly of his. This system is illustrated graphically in Figure 7.3.

Initially, person 1 has a karma of 1, and persons 2 and 3 each have a karma of 5.5. This is because all of the ratings of person 1's documents are 1, and thus each of his documents has an initial rating of 1 according to Equation 7.1 (prior karmas are all equal), and his karma is just the average of his document's ratings, according to Equation 7.2, and therefore is also 1. Each of the four documents authored by person 1 and person 2 has one rating of 1 and one of 10, so its initial rating is $((1 + 10)/2) = 5.5$ according to Equation 7.1. Persons 2 and 3 have a karma of the average of their documents' ratings according to Equation 7.1, and therefore have initial karmas of 5.5.

Consider the next iteration. Person 1's document ratings of 1 remain the same, because they are all equal (1). Person 1's karma also remains 1. However, according to Equations 7.3 and 7.4, person 2's karma now becomes $((((5.5*10)+(1*1))/(5.5+1)) + (((5.5*10) + (1*1))/(5.5+1)))/2) = 8.62$. Person 3's karma is the same as

Figure 7.3: A Model System Consisting of 3 People and 6 Documents, 2 Documents Per Person



person 2's by symmetry.

At iteration 2: Person 1's document ratings and karma are still one. Person 2 and 3 now have a karma of $((((8.62 * 10) + (1 * 1))/(8.62 + 1)) + (((8.62 * 10) + (1 * 1))/(8.62 + 1)))/2) = 9.06$.

After 5 iterations, person 1 still has a karma of 1, and the karma of persons 2 and 3 has risen to 9.1, where it has stabilized asymptotically. Each document written by each person now has a rating equal to the karma of its author (this is not generally true, but is true in this case because of the symmetric nature of the graph). Thus, the effect of the relaxation is to allow persons 2 and 3 to outvote person 1 and have their views dominate the system. If you want a collaborative system to democratically reflect the views of its users, you would want an outcome along these lines.

Next, consider a system with 4 users and 8 documents consisting of two mutual admiration societies of 2 users each, where each user rates the two documents authored by the other member of her society with a 10, and the 4 documents authored by the members of the other society with values of 1. In this system, all karmas and document ratings are set, after the first iteration, to 4, and do not change with further iterations. Thus, there is a standoff.

For my third experiment, I simulated a system in which the distribution of ratings and connections between users and document was random. In this system, document ratings ranged from 1 to 10 and were uniformly distributed across this range (using the computer's standard pseudo-random number generator). There were three parameters in this system: the number of individuals, the maximum number of ratings per individual, and the maximum number of documents authored by a given individual. The latter two parameters are allowed to range randomly from zero to the maximum for each individual, and the actual documents that they rate are also chosen at random. All of these random numbers are again drawn from an even distribution.

I selected a system with 5 individuals, numbered 1 to 5, up to 5 documents authored per individual, and up to 10 documents rated per individual (individuals did not rate documents of which they they were the author). This resulted in 13 documents, numbered one to 13 (thus each individual authored on average 2.6 documents), with a total of 29 ratings made in total, or an average of 2.2 ratings per document. Table 7.1 shows the documents and their authors. Table 7.2 shows all the ratings, giving for each one the document rated, individual doing the rating, and the rating value.

Table 7.1: Documents and Authors for the 5-Individual Simulation

	document												
author	1	2	3	4	5	6	7	8	9	10	11	12	13
1	x												
2		x	x										
3				x	x	x	x						
4								x	x	x	x		
5												x	x

Table 7.3 shows the mean individual karma after 1, 5 and 10 iterations of the recurrence. Table 7.4 shows the mean document rating after 1, 5, and 10 iterations. Convergence is rapid; to two decimal places, by the 5th iteration for both the karma and the ratings. Documents 4, 6, 8 and 12 have only one rating each, so their ratings do not change over time.

Thus, we have seen that it is feasible to build a collaborative system in which both documents and their authors are evaluated.

Table 7.2: Rating Individuals, Documents Rated, and Rating Value for the 5-Individual Simulation

	document												
rater	1	2	3	4	5	6	7	8	9	10	11	12	13
1					1				10		10		
2	9				10		3	9		7		2	6
3	4	1	6						4		9		2
4	4		5	10	9		4						
5	1	4	2		5	5			6	5	1		

Table 7.3: Mean Individual Karma of the 5-Individual System after 1, 5 and 10 Iterations

	Iteration		
User	1	5	10
1	4.50	4.45	4.45
2	3.42	3.39	3.39
3	6.19	6.32	6.32
4	7.08	7.31	7.31
5	3.00	2.70	2.70

Table 7.4: Mean Document Rating for the 5-Individual system after 1, 5, and 10 Iterations

	Iteration		
Doc	1	5	10
1	4.50	4.45	4.45
2	2.50	1.90	1.90
3	4.33	4.89	4.89
4	10.00	10.00	10.00
5	6.25	6.59	6.59
6	5.00	5.00	5.00
7	3.50	3.68	3.68
8	9.00	9.00	9.00
9	6.67	6.38	6.38
10	6.00	6.11	6.11
11	6.67	7.73	7.73
12	2.00	2.00	2.00
13	4.00	3.40	3.40

Next-generation collaborative systems will likely be embedded in virtual environments in which each individual is represented by a three-dimensional constructed "personality" commonly called an avatar. Collaboration environments like this mix the concepts from collaborative filtering with an attempt to build a completely simulated society. If you think about it, a multi-player game where each avatar is controlled by an individual amounts to such a society [61]. Because computer graphics is currently more advanced than re-constructive computer vision, we are likely to see such simulations with avatars before we see systems with virtual environments that actually reflect the real images of people; however, there is quite significant work along the latter lines; see for example, Lanier's work on what he now calls "tele-immersion" [64].

7.4 Review of the Contributions of this Chapter

The main contribution of this chapter is the construction of a relaxation-based system which demonstrates how a decentralized collaborative, democratic system for the filtering of Web documents would work. I show that such a system is feasible, at least for a small, simulated test bed. It extends the current paradigm of the Web with explicit authorship and evaluation (rating) relations, making the Web graph more rich and more informative. Together with other systems of filtering, such as those used by the various search engines and directories, as well as content-based filtering, such an enriched system could filter the Web more effectively. However, such a system would require a high level of participation by its members. In addition, the collection of large numbers of ratings would make the construction of taste groups, in the same manner as GroupLens and MovieLens, a simple matter, by correlating preference vectors and clustering groups of individuals with similar preference vectors.

This system has some similarities to Google, in that they both use a relaxation-based iterative algorithm. However, it differs in that both document authors and documents themselves are made explicit entities in the system; thus the reputations ("karmas") of document authors are also explicit. Ratings are also made explicit, unlike Google, which just uses the Web's linkage graph. As we have seen, this would make the construction of taste groups readily possible as well. Thus a system like this allows for a potentially richer representation of the underlying ontology of the real world of documents, authors, and evaluators than does a system based on link information alone.

Chapter 8

Conclusion: Break out of the Box

8.1 Thesis Summary, Results, and Contributions

Much of this thesis has been a study in the feasibility of combining hyper-link, textual, manual classification, and collaborative user feedback information in order to build better interfaces to the Web, and thereby facilitate the further development of user communities.

In Chapter 3, I examined the extent to which link information can be combined with information of the category of the page that the link is pointing from to determine the category of a destination page. The first contribution of this chapter was the following: by looking at the confusion matrix, I showed that simple counting of direct or indirect (two-hop) links into or out of a page predicts that page's category with a probability usually higher than chance and in many cases considerably so. I showed that this was the case for both a set of Wisconsin policy pages that I had manually classified and a set of Dmoz pages that had been classified by others.

Also in Chapter 3, I examined a set of variants to the Naive Bayes algorithm, some of which were modified so as to use link information. All of these were tested on data drawn from two parts of the Dmoz hierarchy (two training sets and two corresponding test sets). This led to the second main contribution of this chapter: I found that an algorithm that allowed category information garnered from link information to supersede the result of the Naive Bayes algorithm, except when there was no relevant link information (in which Naive Bayes's result was used) performed statistically better than all the other variants on the two test sets. This result has significant practical results for the Web, in which the number of manually-classified pages continues to increase and this pages are much more likely to be hub or authority pages.

In Chapter 4, I experimented with techniques for merging search engines with directories. Here, I argued that rankings within any vertical portal within a directory need to be specific to that portal, and the union of a very large space of such portals contains the same pages that a search engine does, although organized differently. I argued that the ranking of pages within such a vertical portal needs to look at linkages within that portal as well as the goodness-of-fit (quality of set membership) of individual pages within that portal. This goodness-of-fit is a measure of the relevance of the page to the portal's topic. The pages that should be ranked the highest within a portal should be highly relevant pages that are pointed to by a lot of other highly relevant pages within the portal.

The main contribution here is conceptual: that is, pointing out the difference between contextualized and global ranking schemes, and noting that a global rank and a contextualized rank are unlikely to be the same, because there are likely to be pages with high global rank which are only slightly relevant to a particular topic.

Figure 8.1: Hypothetical Optimal Rankings of a Set of Ten Web Pages with Respect to Two Topics

Topic 1	Topic 2
1. Page A	1. Page G
2. Page B	2. Page J
3. Page C	3. Page I
4. Page D	4. Page F
5. Page E	5. Page E
6. Page F	6. Page D
7. Page G	7. Page A
8. Page H	8. Page H
9. Page I	9. Page C
10. Page J	10. Page B

In this case, a page that is highly on-topic, but with a lower global rank, should be ranked higher in a contextualized ranking.

Figure 8.1 illustrates this idea. Here, ten pages are ranked with respect to each of two topics, according to a hypothetical optimal ranking. Being highly ranked with respect to one topic does not mean that a page is highly ranked with respect to the other. For instance, page A is ranked first under topic 1, but seventh under topic 2.

In Chapter 4, to illustrate these concepts, I built a sample system (Active Portal) for collecting pages within particular topics by spidering off of an existing classification. This system was evaluated with respect to a few other directories and with respect to Google, and appeared to hold up well, retrieving more sites than the directories (with, however, less precision) and having higher precision than Google with respect to simple queries (with, however, less recall). However, more work needs to be done to improve the precision of this system, and to adapt its ranking system so that it can predict the ranks of actual users. This would involve the collection of a large amount of human subject data. (Recall is of little concern in the Web context, because people usually have more information than they can handle; it is only relevant in the sense that one wants to be sure not to miss the most important, most salient sites).

Also in Chapter 4, I experimented to see to what extent a multi-resolution version of Naive Bayes can be used in conjunction with spidering in order to classify pages. Again, I used pages from Dmoz as a test bed for this. Looking at pages spidered off of existing pre-classified pages by following links off of those pages, I found levels of classification performance that are considerably higher than one would expect by chance, even though these spidered pages do not necessarily fall into the same category as the page that points to each of them.

Because the performance is quite high at the top level (lowest level of resolution), I found that pages tend not to be categorized not far from the category of the page pointing to them (i.e., in a sister category). Thus this shows that this is a feasible strategy for classifying pages in a tree-like structure, and is a contribution of this chapter. Thus, using a multi-resolution Naive Bayes algorithm to feed pages into vertical portals for consumption and filtering by communities also seems feasible. It would be interesting to implement one or more portals and collect feedback to see to what extent the pages being fed into the portal meet the approval of the community members.

Also in Chapter 4, I experimented with the idea of using the most prominent keywords from the centroid of a set of pre-classified pages to gather new pages using a meta-search. With the two categories that I used in this experiment, I found that this worked quite well. I also found that these pages classified very accurately back into their intended category using Naive Bayes, but this was not that surprising, given the fact that Naive Bayes itself responds the most to the highest-probability keywords. I also, in this same section, explore the differences between using a TFIDF and what I call a TFIGF (term-frequency-inverse-global-frequency) in generating the centroids, and find that a combination of these two approaches appears to work best, at least for these data. The idea, and discovered feasability, of using the keyword centroids to find more pages on a focussed topic is the contribution of this part of the chapter.

In Chapter 5, I explored relevance and quality judgments given by Web users. I argued that the mean relevance and quality of a Web page, as rated by a group of users, is a more meaningful metric of what users think of a page, because it is a direct measure, rather than indirect measures such as schemes to count in-links, even sophisticated recursive schemes such as Google's. I measured the quality and relevance to a category of 500 Web pages, 100 each classified into each of five categories.

I found that the keywords on the pages alone failed to predict my quality and relevance measures with any probability exceeding that that would occur by chance, at least for the models that I tested. I also found that the logarithms of the search engine counts of in-links, as given by two different search engines, weakly but significantly predicted these quality and relevance judgments, accounting for between 10 and 20 percent of the variation in these measures. These negative and positive results are contributions of this chapter.[1]

Chapter 5 showed that a simple count of in-links does a poor job of capturing people's views of what high quality and relevant pages are. It appears that social factors (like how well-known the institution that created the page is) and how well-designed the page is may play a large role. In any case, this chapter points up the need for direct measurement of human preferences, such as in collaborative filtering systems, as opposed to information that is automatically collected.[2]

In Chapter 6, I explored some techniques for clustering Web pages. (There were also a couple of instances of classification undertaken in this chapter, as mentioned below.) First, I looked at the idea of using connected components of pages on the Web to identify pages that are semantically related. This ran up against a problem; sometimes semantically unrelated pages are connected. It turns out that such pages tend to be connected through pages that are very heavily referenced on the Web, such as the Yahoo! home page. It turns out that if you prune such pages out of the graph, this removes the problem, at least for the pages with which I experimented. This was the first contribution of this chapter.

I used this technique to semantically separate pages that result from three ambiguous queries to search engines, and found that it gave good separation. I looked at the two new search engines Teoma and Wisenut, seeing how they did on this semantic separation task, and found that their performance was mixed, doing better on some tasks than others. Overall, I would access their performance as fair. I suggest a tech-

[1]I was unable to get the Google scores, since Google does not give these out directly, in numerical form, but just uses them to rank pages presented to the public; it is possible that the Google scores would predict the quality and relevance scores more accurately.

[2]However, it may be the case that Nielsen-style usage statistics and measures of time spent on a page, and scrolling within it, can be useful proxies.

nique, not yet implemented or tested, for modifying the similarity function between pages to take account of both links between pages and textual similarity.

Also in Chapter 6, I looked at the extent to which WordNet could be used for semantic disambiguation. For two different ambiguous queries, I tested whether the cosine of the WordNet semantic neighborhood, as I defined it in that chapter, of each query and a page in question could successfully determine the meaning of that page, and I found that this worked reasonably well, when you look at the confusion matrices in question. Of course, this only worked when WordNet had the meanings in question in its database. Since WordNet is fixed in structure, and we are matching against it, this was, strictly speaking, classification, not clustering. The demonstration of the feasibility of using WordNet for semantic disambiguation in the manner I suggest was the second contribution of this chapter.

In the final section of Chapter 6, I looked at the performance of HAL-derived semantic networks in clustering sets of Web pages. I considered two methods for using such networks for disambiguating the results of ambiguous Web queries, which I call *HAL-Set-Partition* and *HAL-Disjoint-Supervised*. The former is totally automated, and the second requires some user input. *HAL-Set-Partition* considers the top correlates of the query word (ignoring stop words), computes their correlates, and then connects all words correlated in some way to as to form sets of words. *HAL-Disjoint-Supervised* has the user select salient meanings within the top correlates to the query word, and then forms sets around each of these, eliminating words that appear on more than one list. The second method works better, which is not surprising since it uses user supervision. However, it would be easy to build a system that asks for this supervision. The presentation of the semantic network of words around a query word would be interesting information for many people doing a query, and would be useful for query refinement. *HAL-Disjoint-Supervised* is a supervised algorithm, so it is a classification rather than a clustering technique. The development of these two techniques, especially the more effective *HAL-Disjoint-Supervised,* is the third contribution of this chapter.

In Chapter 7, I discussed a number of emerging systems for collaboration among users. As I will argue in the remainder of this conclusion, I believe such collaboration will be central to successful systems on the Web in the future; the best systems will combine techniques of collaborative filtering and automatic filtering, playing to the strengths of man and machine. The machine is best at processing large amounts of information and at doing a "first cut" at filtering that information; once this first cut has been done, then collaboration and individual editorial work is best at filtering it for final consumption.

Chapter 7 first discussed a number of systems that have been developed in order to overcome the signal-to-noise problems of earlier systems such as Usenet News. Usenet News initially benefitted from the self-selection of more adept users in the earlier years of the Internet (that is, when the Internet was still mainly used by a small elite in universities and other elite institutions). Later, it became flooded with spammers and low-quality messages from uninformed users. As a result, people developed a type of system that allowed people to develop reputations in the system, and to moderate the system. People were rated on the quality of their comments, and this became the basis of their reputations.

I argued in Chapter 7 that document ratings and user reputations that are based on these document ratings are an important piece of meta-data that should be inserted, if possible, in the next generation of the Web. The authorship relation between a person (or institution) and a document is also an important piece of meta-data. (These are among many such semantic relations that should be recorded in meta-data.) I built a model of a iterative linear system in which users rate documents that they are not the author of, and develop reputations based on the ratings of the document. Their ratings are then weighted by their reputations, and the system relaxes through iterations to stable values for the user reputations and overall document ratings. I ran several variants of this model and show that they all converge. The purpose of this exercise was to design a system that could be implemented as a component of an overall system for document filtering and collaboration, a system that also made use of multi-resolution automatic classification and of clustering. The "proof-of-concept" provided by showing the feasability of this relaxation-based system for embodying user ratings and user document ratings was the main contribution of this chapter.

8.2 Directions for Further Research

At various points in this thesis, I have suggested possibilities for further research. I review these possibilities briefly in what follows here, as well as some ideas that were not already mentioned.

My work in Chapter 3 on the automatic classification of Web pages suggests the design of a system that feeds newly-discovered Web pages to users for feedback and classification, allowing training sets to expand in a semi-automated fashion. The system could use the most successful algorithm in Chapter 3 (*Voting-Trumps-NB*) to make a provisional classification of a new Web pages. It could feed Web pages to users for classification in a priority order based on the number of links from pages already known to be a particular category. Users could confirm or reject the machine's classification, allowing the training set to expand with a reduced cognitive load on the community of users. As the training set enlarges, the machine's classification performance would presumably get more accurate. Such a system could also function as a semi-automated tool for the construction of taxonomies such as those now constructed manually by Yahoo! and Dmoz.

In addition, additional work could be done on the design of algorithms that use link information in addition to textual information in classifying Web pages. I found that *Voting-Trumps-NB* performed significantly better than Naive Bayes on the testbeds that I was using, but it needs to be tested on additional testbeds. In addition, it needs to be compared with non-Naive-Bayes-based algorithms, some of which are known to perform better on the text classification task than Naive Bayes (see Section 2.10 for a discussion of this). Given that the addition of the link information to Naive Bayes in one form is improving its performance, presumably the performance of these other, better-performing algorithms could also be improved. However, this needs to be validated experimentally, and ways need to be devised to incorporate the use of the link information in these algorithms.

Other algorithms could be devised that also make use of the anchor text that appears in Web page links; this often contains a useful summarization of the text of the page that it points to. This anchor text has been used by Blum and Mitchell in their work on "co-training" [11].

An important point made in this thesis that the rankings of pages with respect to a Web query, or with respect to a particular topic, need to be specific to a particular topic. For instance, a page that should be ranked highly with respect to one topic, because it contains a large quantity of high-quality information about that topic, should be well down the list with respect to another topic, if it only touches on this latter topic. The search engines, as far as I know, have not collected much human subject data in an attempt to make their rankings fit the rank preferences of human subjects. If such subject data is collected, it would be possible to compare different ranking schemes. This seems like an important area of research if search engines are to continue to be improved. Chapter 5 found that subjects' quality and relevance judgments were not very strongly related to search engine rankings as they presently stand, so this is indicates that there is substantial room for improvement in this area. The AI challenge here is substantial, to build systems that can recognize high-quality information (as judged by humans).This is presumably related to text comprehension, which is known to be a difficult problem.

Specifically, I suggest in Chapter 3 that ranking schemes can be improved by taking into account topical information. A page with in-links from pages that are *already known to be on the topic in question* should be ranked higher than those without such inlinks, all else being equal. However, I have not fully validated this idea with data; collection of subject feedback data on particular rankings with respect to particular topics could do so. Additional work needs to be done to better fit these subject ratings, so that they can be better predicted. In addition, work needs to be done to model the entire search process, from query formation to the presentation of results, and then collecting feedback on the quality of the overall results. This is related to work on how to build systems that help users automatically improve their queries.

The sections of Chapter 4 on multi-resolution Naive Bayes classification and on using the centroid of set of Web pages on a particular topic to generate keywords to pass to search engines to find additional pages on this topic can be combined to devise a system for automatically classifying Web pages. New pages are fed into the system via spidering and via the output of these search engines and then classified using multi-resolution Naive Bayes.

Further work could also be done on identifying keywords that work well in defining topics. I have begun this work with my work on the computation of the TFIDF centroid and what I call the TFIGF centroid, and the combined version of this, but you could imagine a more extensive algorithm that inductively (based on a training set of classified pages) builds a query string that uses phrases, conjunctions, and disjunctions to more accurately pull out pages on a particular topic. Words from the TFIDF or TFIGF centroids could be used to seed these potential query strings, and alternative query strings could be tested against one another in terms of their precision and recall in discovering members of the training set. One could also look for the most-frequently-occurring two-word or longer strings in the positive training examples.

In order to do this, techniques that combine machine learning and logical representations of query strings could be used, such as inductive logic programming [79]. The output of these queries could be fed back into the system for user feedback, allowing for continuous improvement of a system that identifies pages on particular topics; the query strings could be continuously relearned.

The technique in Chapter 6 that clusters Web pages using connected components works fairly well, using link information alone, but it could be enhanced by using it in combination with a standard text-based clustering algorithm. The connected components could be used to create the initial clusters, but text-based methods could be used to combine clusters if they were sufficiently similar. Alternately, the document similarity function itself could be altered to take both link and textual information into account. These ideas have been discussed in more detail in Section 6.1.7.

In Chapter 6, I applied WordNet to the problem of semantic disambiguation of Web pages referrring to different meanings of the same word. WordNet is a wonderful tool, and researchers have only scratched the surface of its applicability to problems involving text management on the Web. For instance, it could also be applied to query refinement, after it has made an initial stab at disambiguation of the results of a Web query. Pages could be separated into several topics corresponding to different meanings of a search term in WordNet, and then additional words associated with each meaning could be suggested in order to construct one refined query for each of the possible meanings of the initial one-word query.

HAL-based semantic networks were also used for clustering in Chapter 6. The two algorithms developed in that chapter along these lines, *HAL-Set-Partition* and *HAL-Disjoint-Supervised,* both had certain flaws: namely, *HAL-Set-Partition* did not work that well, and *HAL-Disjoint-Supervised* requires user supervision. Further work could be done to develop an algorithm for clustering based on such semantic networks that works better than *HAL-Set-Partition* without having to resort to user supervision. If a method could be arrived at for automatically identifying the central keywords that are provided by the user in *HAL-Disjoint-Supervised*, then this algorithm would no longer require supervision. However, identifying these central words may be a very difficult task, in the absence of outside information.

The system for collaborative filtering that I described in Section 7.3 needs to be tested on real-world data. Ideally, a situation could be set up where people would get some sort of reward for achieving high karma in the system. Building a real system and deploying it would test it with a much larger number of authors and documents and real-world situations and most certainly reveal aspects of it that need to be redesigned. Actual users may not view the system for assigning karmas and document ratings to be fair and may demand another one.

8.3 From Many Boxes to One

One of the main ideas of this thesis can be stated in terms of the old saw: “think outside the box.” Despite the Web being less than a decade old, we have already circumscribed the types of systems that exist on it: for instance, ordinary Web sites, search engines, directories, collaborative filtering sites, discussion forums, and auction sites. These are all “boxes” that constrain our thinking. The next generation of tools needs to combine many of these ideas into single, integrated systems. For instance, there is little difference in principle between a list of bookmarks (which represent a single user’s preferences), a directory system such as Dmoz or Yahoo!, which represent the aggregated knowledge and preferences of a set of user/editors, a collaborative filtering system such as the IMDB, MovieLens, or Slashdot, which also represent a set of preferences, and a search engine, which adds a usual text index to a ranking system which reflects preferences, typically in terms of the number in-links that a page receives. All of these represent preferences in one way or another, and therefore can be integrated into a single system which combines all of their features, allowing the user to visualize information and participate in communities in a highly flexible manner. All of them represent instances of the general concept of *information filtering*, which needs to be at the core of any successful Web system, although not necessarily its only component.

Traditionally, information filtering has been a task undertaken by the press and the publishing/media establishment in general. This operates on a bottom-up followed by top-down model; authors and journalists submit their articles through an administrative process controlled by editors, who filter these articles, rejecting some of them, and accepting and typically modifying others. Then, mass marketing and branding is used to deliver this information over channels that are often quite oligopolistic. The Web has the potential to change all this, although it is not clear whether or not it will, because this model has managed to transfer over to the Web domain without many changes—the major media outlets now all have Web sites, many of which are quite successful.

Nevertheless, the Web has vastly lowered the barriers to entry for anyone wanting to communicate with the rest of the world or with a specific audience; an individual can easily start a Web site, although building an audience can be difficult. Some people with specific interests have already become modestly successful in building "human portals" [16] that can compete, on some level, for audience with the mass media sties. Many of these sites feature Web logs, or "blogs", whose creation has become easy due to the wide dissemination of software called "Blogger" (see www.blogger.com). A Web log is a kind of electronic journal with links. People look at these human portals and follow the blogs of their authors. A blog could also be thought of as an annotated bookmark list. As in a collaborative filtering system, such a document represents, on some level, the preferences and interests of its author.

In this thesis, I have discussed a variety of ideas that are common at the convergence of the fields of machine learning, information retrieval, and their application to the Web domain. These have included page classification, page clustering, inverted-index search engines, and collaborative filtering.

One of the main ideas of this thesis is that there has been an artificial distinction made between Web inverted-index search engines and Web directories. These two things would work better of as two aspects of the same system. It is possible to use an algorithm such as a multi-resolution version of the Naive Bayes algorithm to automatically classify all Web pages as they are encountered by a spider. Thus, the directory and the search engine become just two different ways to look at the same set of pages, with linkages from each page listed in search engine results to the category of which it is a member.

In addition, the ordering of pages within a category can be specific to that category, rather than using a global ranking such as Google's PageRank. This ordering can be based on linkages within the category (the page having more links from other pages in the category is ranked higher), or the number of words characteristic of a category that a page has (normalized by its length), or a combination of these. Contextualized rankings are superior to non-contextualized ones because they only take account of category-specific information; global information is of little use to searchers looking for topical information.

Also, there has been an artificial distinction made between collaborative filtering tools and search engines and directories. A directory is simply a filtering system without the rating attached to each page (since multiple people typically work on each directory, which is typically so huge that one person cannot produce it—it requires a whole team). A search engine can be thought of as a collaborative filtering system, in that it filters information based on its content and, typically, in terms of in-links, which themselves are created by groups of people. Making a link is a way to filter. Putting something into a directory is a way to filter.

System architects are energetically planning the next generation of tools that will be more useful to individuals looking for information, and communities interacting with information (and I argue that the latter is a better way to think about the search process[3]). I argue in this thesis that the best such tools need to involve all of the techniques discussed herein combined into one, single, simple, integrated system.

Consider what I will call, for lack of a better term, a collective search and sharing system. In such a system, any one may choose to participate in one or more communities on the system. Information items, which may include Web pages, formatted documents such as Adobe Acrobat PDF files, music files, works of art, etc. will be full-text indexed in the system as much as feasible (image indexing, for instance, may be difficult). Reverse-keyword searches, as in current search engines, will of course still be possible. But the system will also make an effort to cluster and classify all of the information items of which it is aware.

In addition, communities of users within the system with an interest in particular information items will participate in the collaborative filtering of such items. Each community will be explicitly identified and trust and prestige filtered in the network,

[3]The American tendency toward individualism has to some extent blinded us to the usefulness of community interaction in collective endeavors involving information, whether these endeavors are, for example, science, writing, or studying specific academic subjects.

as well as the quality of the documents themselves. Thus documents will get ratings and users, reputations. Items will be contributed to each "community portal" within the system either explicitly by the users of that portal or automatically by spiders that pass their new resources not only to the usual full-text indexer but also to classifiers or clustering algorithms that decide which portal(s) in which each resource should be placed. These automatically-gathered resources, along with the manually-added ones, are available for collaborative filtering by the users of the system. In addition, the system may attempt to predict the collaborative filtered score(s) for each document based on document characteristics and its place in the link structure of the Web. Collaborative filtering and automatic techniques can create a synergy, where a combined system is more than the sum of its parts. There can be multiple competing portals on a single topic, but each will probably mine links from the other.

Thus, the result of a Web search will be quite different than what we see today. Instead of the isolated searcher, the searcher automatically sees her results grouped into communities of interest with active participants, along with scores that those participants have assigned to the content in that community. She can then directly go to the portal of one of these communities, sign up as a member, and start participating, or simply view the results as a guest. As she gets more involved with one of these communities, she can build up her reputation in the community through effort, and engage with the community in collaborative or individual document creation (where a document could be a written document, a computer program, a lecture for a class, a design for a building, a piece of artwork, or a piece of music). These documents would then be filtered by the community, and organized by the classification, clustering, and full-text search software. In some cases, there would a market aspect to this; people would create documents that other people would purchase (such as companies who need software written, or a person who wants a house designed). As we can see, the social potential for such systems is endless.

8.4 Thinking Even More Broadly: Transforming Society through Collaborative Information Management and Creation

One can conceive of collaborative activities more even broadly than collaborative communities involving information. Essentially, the economy, and society as a whole, is a large collaborative system. The price mechanism in the economy is far from perfect, because of such problems as asymmetric information [2], adverse selection [91], monopolies, and bounded rationality [113]. To cope with such problems, firms, workers, and consumers have relied on critics and impartial sources of information (e.g. the Zagat restaurant guide or Consumer Reports) and such strategies as branding (e.g., in consumer electronics, Sony is viewed as a higher-end and therefore a higher-quality brand than, say, Emerson). Firms routinely attempt to gather more detailed feedback on their performance than is given by the simple price signal, for example by doing satisfaction surveys of their customers on a variety of parameters.

Collaborative filtering systems have the potential to make a major impact on the economy. The well-known eBay auction site (at ebay.com) has sellers and buyers rate each other; if a user's rating, whether they are a seller or buyer, goes too low, they are kicked off the system. Provided that a mechanism is created for preserving identities and reputations—in itself not a mean engineering feat, both technically and socially—such a system has the potential of solving problems that occur when the market "game" is only played once.

If a model such as this was extended to the economy as a whole, including the labor market, markets for consumer commodities, markets for services such as health care, education, financial, and legal services, and internal markets within supply chains, it could have the potential of revolutionizing society by reducing the many information problems with which markets are plagued. Markets would start to function much more efficiently than they presently do. In addition, the types of learning communities that presently exist in such specialized economic regions such as Silicon Valley, Hollywood, and furniture production in Denmark, and in many realms of science and the arts, might flourish through virtual means [73, 102].

Of course, such systems have to have high levels of participation to function, and creating the incentives for such participation is difficult, given free-rider problems in which users gain the fruits of participation without fully participating [46]. It is possible that studying techniques from survey research may improve participation (see, for example, [36]). I suspect that increasing user identification with the community would be even more effective; of course, the best thing would be to do both.

Another possibility is to start to shift the incentive system, since people presently have little incentive to participate in online activities that have little relation to their gainful employment. Presently, there are two major disincentives to online publishing. The first is that, unlike publishing on paper, Web publishing has only had mild positive incentives, which are money from advertising and the fruits of participating in an online community.[4] Only occasionally have people paid cash for individual electronic documents, and most documents are not available, because of piracy concerns. Publishing on paper is still more profitable, because you can sell the published work; companies have not solved the problem of how to sell things electronically without enormous losses through unauthorized copying, and this remains a huge, unsolved problem: how to compensate people for their work on the Internet. The second is that people do not like to read at their computer, because the screen has lower contrast and resolution than paper, and has a worse interface (typically scrolling rather than the preferred codex.) The second of these may be solved before long through the development of wireless electronic "paper" "books" [31]. In addition, as broadband increases, the possibilities for face-to-face interaction through video-phone or video-conferencing will remove some of the impersonal feel of the Internet. However, the incentives for the deployment of broadband remain weakened as long as the legal stalemate on the electronic dissemination of media continues.

Probably more significant than these disincentives is the social inertia of institutions, which is largely due to the entrenched distribution of power. Consider academia. For students, the incentive system is based on letter grades, for the most part (although faculty recommendations and some non-academic factors also play a role, such as community service and athletic participation). For faculty, the main incentives are quantity and quality of research, and the quality of teaching, weighted appropriately depending on the type of institution.

One could conceive of a new kind of educational institution in which grades were assigned as a result of participation in collaborative learning communities, and re-

[4]And advertising also lowers participation, by "polluting" the net with distractions.

search was accessed through similar communities of researchers. There would be less of a bright line between students and teachers; many people would act in both roles. Grades would be assigned by the collective assessment of the community. People would be compensated based on the degree to which they contributed to the learning of others, as measured by the collaborative community itself, and the degree to which their writing and research contributed to the overall base of knowledge, again as assessed by the community. Students would pay for the system (perhaps with a government or endowed subsidy), and faculty would get payments, but some people would play both roles concurrently. Of course, moving to such a system of governance and incentives is a radical shift from the present arrangements, and as such is likely to create resistance. All of the online universities that I know of maintain a sharp line between faculty and students, although some allow peer assessment among the students. And online universities have not made much of an impact, in general. The legal issues here are complex, since some students may object to being evaluated to anyone other than the faculty.

Or consider firms in the economy. Many have looked to the open source movement in the software development community as a new model of how people might work together to create (for more thoughts on this, see, for instance, [68, 69, 96]). There are tools such as SourceForge (at www.sourceforge.net) which developers use to develop software together. But many other types of knowledge workers, such as lawyers, doctors, accountants and financial professionals of all stripes, architects and other visual designers, artists, musicians, and engineers, could potentially make use of the Open Source model, "ported" to their field. Again, collaborative communities would access the work, and customers for the work would pay for it. Traditional hierarchies within firms would play little role– instead, workers would be sought on the basis of their proven ability to contribute to projects.

Collaborative systems that manage information well and help people create new ideas, using all of the best qualities of people and machines, have the potential for great enrichment of the human experience. Such systems tend to be relatively non-commercial, democratic, merit-based, and decentralized. Contrast this with the institutions that we have today, which are often overly commercial, non-democratic, sometimes not based on merit (but, for instance, on "who you know" or on how good or extensive one's marketing and public relations is), and hierarchical. It appears to me that the near future will involve a choice between these two types of institutions, and it is not at all clear what will be chosen. Commercial interests are working hard at building a commercially-based next-generation Internet, which will probably look like some sort of combination of America Online and cable television. But further work on creating alternative collaborative systems and making them function as well and as fully as possible—which is just as much a problem of human and social engineering as it is a technical problem—will improve the chances that such alternative systems will be embraced by the public.

Bibliography

[1] Charu C. Aggarwal, Stephen C. Gates, and Philip S. Yu. On the merits of building categorization systems by supervised clustering. In *Proceedings of KDD-99, 5th ACM International Conference on Knowledge Discovery and Data Mining*, pages 352–356, San Diego, US, 1999. ACM Press, New York, US.

[2] G. Akerlof. The market for lemons: Quality uncertainty and the market mechanism. *Quarterly Journal of Economics*, 84:488–500, 1970.

[3] Reka Albert, Hawoong Jeong, and Albert-Laszlo Barabasi. Diameter of the World Wide Web. *Nature*, 401, 1999.

[4] J. Anderson. *The Adaptive Character of Thought.* Lawrence Erlbaum, Hillsdale, NJ, 1990.

[5] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrival.* ACM Press/Addison-Wesley, New York, 1999.

[6] Amit Bagga, Joyce Y. Chai, and Alan W. Biermann. The role of WordNet in the creation of a trainable message understanding system. In Howard Shrobe and Ted Senator, editors, *Proceedings of the Thirteenth National Conference*

on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, pages 941–948, Menlo Park, California, 1996. AAAI Press.

[7] Brian T. Bartell, Garrison W. Cottrell, and Richard K. Belew. Latent Semantic Indexing is an optimal special case of multidimensional scaling. In *Research and Development in Information Retrieval*, pages 161–167, 1992.

[8] R. Belew and J. Hatton. RAVE reviews: Acquiring relevance assessment from multiple users, 1996.

[9] Richard K. Belew. *Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW.* Cambridge University Press, New York, 2001.

[10] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic Web. *Scientific American*, May 2001.

[11] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1998.

[12] T. Bray, J. Paoli, and C. Sperberg-MacQueen. Extensible markup language. Technical report, W3 Consortium, 1998.

[13] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM (1)*, pages 126–134, 1999.

[14] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual search engine. In *Proceedings of the Seventh International World Wide Web Conference*, 1998.

[15] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhaker Raghavan, Sridhar Rajogopalan, Raymie Stata, Andrew Tompkins, and Janet Wiener. Graph structure in the Web. In *Proceedings of the Ninth International World Wide Web Conference*, 2000.

[16] Austin Bunn. Human portals. *Brill's Content*, 4(4):70+, 2001.

[17] C. Burgess, K. Livesay, and K. Lund. Explorations in context space: Words, sentences, discourse. *Discourse Processes*, 25:211–257, 1998.

[18] Steve Burnett and Stephen Paine. *RSA Security's Official Guide to Cryptography.* McGrawHill Osborne Media, Berkeley CA, 2001.

[19] M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, pages 6–11, Menlo Park, CA, 1998. AAAI Press.

[20] F. Can and E. Ozkarahan. Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. *ACM Transactions on Database Systems*, 15(4):483–517, 1990.

[21] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Hypersearching the web. *Scientific American*, June 1999.

[22] Soumen Chakrabarti, Byron E. Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In Laura M. Haas and Ashutosh Tiwary, editors, *Proceedings of SIGMOD-98, ACM International Conference on Management of Data*, pages 307–318, Seattle, US, 1998. ACM Press, New York, US.

[23] Soumen Chakrabarti, Byron E. Dom, S. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins, David Gibson, and Jon Kleinberg. Mining the Web's link structure. *Computer*, 32(8):60–67, 1999.

[24] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom M. Mitchell, Kamal Nigam, and Sean Slattery. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, 118(1-2):69–113, 2000.

[25] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 509–516, Madison, US, 1998. AAAI Press, Menlo Park, US.

[26] Douglass R. Cutting, David Karger, and Jan Pedersen. Constant interaction-time Scatter/Gather browsing of very large document collections. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 126–135, 1993.

[27] Douglass R. Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections.

In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[28] B. Dahlen, J. Konstan, J. Herlocker, N. Good, A. Borchers, and J. Riedl. Jump-starting MovieLens: User benefits of starting a collaborative filtering system with dead data. Technical Report TR-98-017, University of Minnesota, 1998.

[29] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[30] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[31] Steve Ditlea. The electronic paper chase. *Scientific American*, November 2001.

[32] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of ACM-CIKM98*, pages 148–155, 1998.

[33] P. Edwards, D. Bayer, C. L. Green, and T. R. Payne. Experience with learning agents which manage Internet-based information. In M. A. Hearst and H. Hirsh, editors, *AAAI 1996 Stanford Spring Symposium on Machine Learning in Information Access*, pages 31–40. AAAI, 1996.

[34] Ivor H. Evans. *Brewer's Dictionary of Phrase and Fable.* Harper and Row, New York, 1989.

[35] Christiane Fellbaum, editor. *WordNet : an electronic lexical database.* MIT Press, Cambridge, Mass., 1998.

[36] Richard J. Fox, Melvin R. Crask, and Jonghoon Kim. Mail survey response rate: A meta-analysis of selected techniques indusing response. *Public Opinion Quarterly*, 52(4):467–491, 1988.

[37] E. Frank, G. Paynter, I. Witten, C. Gutwin, and C. Nevill-Manning. Domain-specific learning algorithms for keyphrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 668–673. Morgan Kaufmann, 1999.

[38] Malcolm Gladwell. *The Tipping Point: How Little Things Can Make a Big Difference.* Little Brown, Boston, 2000.

[39] J. Gleick. *Chaos - Making a New Science.* Viking, New York, US, 1987.

[40] Jeremy Goecks and Jude W. Shavlik. Learning users' interests by unobtrusively observing their normal behavior. In *Intelligent User Interfaces*, pages 129–132, 2000.

[41] Moises Goldszmidt and Mehran Sahami. A probabilistic approach to full-text document clustering. Technical Report ITAD-433-MS-98-044, SRI International, 1998.

[42] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press and North Oxford Academic, Baltimore, MD, USA and Oxford, England, 1983.

[43] J. Gomez-Hidalgo and M. Rodriguez. Integrating a lexical database and a training collection for text categorization. In *Proceedings of the Workshop in Automatic Information Extraction and Building of Lexical Semantic Resources*, pages 39–44, 1997.

[44] Antonietta Grasso, Jean-Luc Meunier, and Christopher Thompson. Augmenting recommender systems by embedding interfaces into practices. In *Proc. Intl. Conf. On Supporting Group Work (GROUP'99)*, pages 267–275, 2000.

[45] Philip Greenspun. *Philip and Alex's Guide to Web Publishing.* Morgan Kaufmann, San Francisco, 1999.

[46] S. Grossman and O. Hart. Takeover bids, the free-rider problem, and the theory of the corporation. *Bell Journal of Economics*, 11, 1980.

[47] Lawrence C. Hamilton. *Regression with Graphics: A Second Course in Applied Statistics.* Duxbury Press, Belmont, California, 1992.

[48] Marti A. Hearst. Automated discovery of WordNet relations. In Christiane Fellbaum, editor, *WordNet : an electronic lexical database.* MIT Press, Cambridge, Mass., 1998.

[49] Marti A. Hearst. Untangling text data mining. In *Proceedings of ACL'99: the 37th Annual Meeting of the Association for Computational Linguistics*, 1999.

[50] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, CA, 1991.

[51] D. Hill. A vector clustering technique. In Samuelson, editor, *Mechanized Information Storage, Retrieval and Dissemination*. North-Holland, 1968.

[52] T. Howes and M. Smith. *LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol*. Macmillan Technical Publishing, 1997.

[53] Bernardo A. Huberman and Lada A. Adamic. Growth dynamics of the World Wide Web. *Nature*, 401, 1999.

[54] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of ROCLING X*, 1997.

[55] Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *International Conference on Machine Learning*, pages 143–151, 1997.

[56] G. Karypis and E. Sam. Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval and categorization. Technical Report TR-00-0016, University of Minnesota, 2000.

[57] Amy Jo Kim. *Community Building on the Web: Secret Strategies for Successful Online Communities*. Peachpit Press, Berkeley CA, 2000.

[58] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth ACM-SIAM Symposium on Discrete Algorithms*, 1999.

[59] Donald E. Knuth. Dynamic Huffman coding. *J. Algorithms*, 6(2):163–180, 1985.

[60] Yannis Labrou and Timothy W. Finin. Yahoo! as an ontology: Using Yahoo! categories to describe documents. In *Proceedings of the Eighth International Conference on Information Knowledge Management*, pages 180–187, 1999.

[61] John E. Laird and Michael van Lent. Human-level AI's killer application: Interactive computer games. *AI Magazine*, 22(2):15–25, 2001.

[62] George Lakoff and Mark Johnson. *Metaphors We Live By*. University of Chicago Press, Chicago, 1980.

[63] Ken Lang. NewsWeeder: Learning to filter Netnews. In *International Conference on Machine Learning*, pages 331–339, 1995.

[64] Jaron Lanier. Virtually there: Three-dimensional tele-immersion may eventually bring the world to your desk. *Scientific American*, April 2001.

[65] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, pages 16–22, New York, 1999. ACM Press.

[66] Steve Lawrence and C. Lee Giles. Searching the World Wide Web. *Science*, 280(5360):98–100, 1998.

[67] Douglas B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[68] Lawrence Lessig. *Code: And other Laws of Cyberspace*. Basic Books, New York, 2000.

[69] Lawrence Lessig. *The Future of Ideas*. Random House, New York, 2001.

[70] David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 298–306, Zürich, CH, 1996. ACM Press, New York, US.

[71] Xiaobin Li, Stan Szpakowicz, and Stan Matwin. A WordNet-based algorithm for word sense disambiguation. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal*, pages 1368–1374, 1995.

[72] Benoit Mandelbrot. *The Fractal Geometry of Nature.* W.H. Freeman, New York, 1983.

[73] Alfred Marshall. *Principles of Economics.* Macmillan, London, [1920] 1961.

[74] Briji Masand, Gordon Linoff, and David Waltz. Classifying news stories using memory-based reasoning. In Nicholas J. Belkin, Peter Ingwersen, and Annelise Mark Pejtersen, editors, *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 59–65, Kobenhavn, DK, 1992. ACM Press, New York, US.

[75] R. Mcarthur and P.D. Bruza. The ranking of query refinements in interactive web-based retrieval. In *Proceedings of Information Doors: Where Information Search and Hypertext Interlink*, 2000.

[76] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[77] Robert W. McChesney. The Internet and U.S. communication policymaking. *Journal of Computer-Mediated Communication*, 1(4), 1996.

[78] George Miller. Ambiguous words. *Information Impacts Magazine*, March 2001.

[79] Tom M. Mitchell. *Machine Learning.* McGrawHill, New York, 1997.

[80] Dunja Mladenic. Turning Yahoo! into an automatic Web-page classifier. In *European Conference on Artificial Intelligence*, pages 473–474, 1998.

[81] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of DL-00, 5th ACM Conference on Digital Libraries*, pages 195–204, San Antonio, US, 2000. ACM Press, New York, US.

[82] Andrew W. Moore and Mary S. Lee. Efficient algorithms for minimizing cross validation error. In *International Conference on Machine Learning*, pages 190–198, 1994.

[83] I. Moulinier. A framework for comparing text categorization approaches. Technical report, LAFORIA-IBP-CNRS, University of Paris, 1996.

[84] Theodor Holm Nelson. The unfinished revolution and Xanadu. *ACM Computing Surveys (CSUR)*, 31(4es):37, 1999.

[85] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 792–799, Madison, US, 1998. AAAI Press, Menlo Park, US.

[86] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[87] Michael J. Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill and Webert: Identifying interesting Web sites. In *AAAI/IAAI, Vol. 1*, pages 54–61, 1996.

[88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, San Mateo, California, 1988.

[89] Peter Pirolli. Computational models of information scent-following in a very large browsable text collection. In *Proceedings of the CHI '97 Conference*, pages 3–10, 1997.

[90] Peter Pirolli, James Pitkow, and Ramana Rao. Silk from a sow's ear: Extracting usable structures from the web. In *Proc. ACM Conf. Human Factors in Computing Systems, CHI.* ACM Press, 1996.

[91] E. Prescott and R. Townsend. Pareto optima and competitive equilibria with adverse selection and moral hazard. *Econometrica*, 52:21–45, 1984.

[92] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge, UK, 1986.

[93] J. Quinlan. *C4.5: Programs for machine learning.* Morgan Kaufmann, San Mateo CA, 1993.

[94] J. Ross Quinlan and R. Mike Cameron-Jones. Induction of logic programs: FOIL and related systems. *New Generation Computing*, 13:287–312, 1995.

[95] E. Rasmussen. Clustering algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 419–442. Prentice-Hall, 1992.

[96] Eric S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary.* O'Reilly, Sebastopol, CA, 2001.

[97] R. Richardson and A. F. Smeaton. Using WordNet in a knowledge-based approach to information retrieval. Technical Report CA-0196, School of Computer Applications, Dublin City University, Ireland, 1995.

[98] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the ASIS*, 27:29–146, 1976.

[99] J. Rocchio. *Document Retrieval Systems–Optimization and Evaluation.* PhD thesis, Harvard University, 1966.

[100] D. Rumelhart and D. Zipser. Feature discovery by competitive learning. *Cognitive Science*, 9:75–112, 1995.

[101] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. I: Foundations*, pages 318–362. Bradford Books/MIT Press, Cambridge, MA., 1986.

[102] Charles F. Sabel. Learning by monitoring: The institutions of economic development. In Neil J. Smelser and Richard Swedberg, editors, *The Handbook of Economic Sociology*. Princeton University Press, Princeton NJ, 1994.

[103] M. Sahami, M. Hearst, and E. Saund. Applying the multiple cause mixture model to text categorization. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 435–443. Morgan Kaufmann, 1996.

[104] Mehran Sahami, Salim Yusufali, and Michelle Q. W. Baldonado. SONIA: A service for organizing networked information autonomously. In *Proceedings of the Annual Conference on Digital Libraries*, 1998.

[105] G. Salton, J. Allan, and C. Buckley. Automatic structuring and retrieval of large text files. *Communications of the ACM*, 17(2):97–108, 1994.

[106] G. Salton and A. Wong. Generation and search of clustered files. *ACM Transactions on Data Base Systems*, 3(4):321–346, 1978.

[107] Gerard Salton, editor. *The SMART retrieval system: experiments in automatic document processing*. Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

[108] Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jonathan L. Herlocker, Bradley N. Miller, and John Riedl. Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system. In *Computer Supported Cooperative Work*, pages 345–354, 1998.

[109] Eric Saund. A multiple cause mixture model for unsupervised learning. *Neural Computation*, 7(1):51–71, 1995.

[110] Amon Seagull. A compaction of WordNet senses for evaluation of word sense disambiguators. Technical Report TR726, University of Rochester, 2000.

[111] C.E. Shannon and W. Weaver. *The Mathematical Theory of Communication.* University of Illinois Press, Urbana IL, 1949.

[112] Jude W. Shavlik, Susan Calcari, Tina Eliassi-Rad, and Jack Solock. An instructable, adaptive interface for discovering and monitoring information on the World-Wide Web. In *Intelligent User Interfaces*, pages 157–160, 1999.

[113] Herbert A. Simon. *Models of Bounded Rationality, Volume 1.* The MIT Press, Cambridge, Massachusetts, 1982.

[114] Munindar P. Singh, Bin Yu, and Mahadevan Venkatraman. Community-based service location. *Communications of the ACM*, 44(4):49–54, 2001.

[115] Sean Slattery. Unsupervised structural inference for Web page classification. Technical report, Carnegie Mellon University, Computer Science Department, 1999.

[116] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.

[117] Mark A. Stairmand. Textual context analysis for information retrieval. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 140–147. ACM Press, 1997.

[118] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on Web-page clustering. In *Proceedings of the 17th National*

Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search (AAAI 2000), 30-31 July 2000, Austin, Texas, USA, pages 58–64. AAAI, July 2000.

[119] Loren G. Terveen and William C. Hill. Evaluating emergent collaboration on the Web. In *Computer Supported Cooperative Work*, pages 355–362, 1998.

[120] E. Voorhees. Implementing agglomerative hierarchical clustering algorithms for use in document retrieval. *Information Processing and Management*, 22:465–476, 1986.

[121] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[122] A. Weigend, E. Wiener, and J. Pedersen. Exploiting hierarchy in text categorisation. *Information Retrieval*, 1(3):193–216, 1999.

[123] Peter Weiner. Linear pattern matching algorithms. In *IEEE Symposium on Foundations of Computer Science*, pages 1–11, 1973.

[124] E. Wiener, J. Pedersen, and A. Weigend. A neural network approach to topic spotting. In *Symposium on Document Analysis and Information Retrieval*, 1995.

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

[126] W. Wong and A. Fu. Incremental document clustering for Web page classification. In *IEEE 2000 Int. Conf. on Info. Society in the 21st century: emerging technologies and new challenges (IS2000), Nov 5-8, 2000, Japan*, 2000.

[127] Y. Yang and X. Liu. A re-examination of text categorization methods. In *22nd Annual International SIGIR*, pages 42–49, Berkley, August 1999.

[128] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69–90, 1999.

[129] Lofti A. Zadeh. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 2:103–111, 1996.

[130] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.

[131] Lisa Zeidenberg. Comparison of Web directory performance: Active Portal, Yahoo, Look Smart, Snap, and About. San Jose State University paper for Library Science 344: Online Searching, August 2000.

[132] Shoshana Zuboff. *In the Age of the Smart Machine.* Basic Books, New York, 1988.