

Conceptual Indexing: A Better Way to Organize Knowledge

William A. Woods

© Copyright 1997 Sun Microsystems, Inc. The SML Technical Report Series is published by Sun Microsystems Laboratories, a division of Sun Microsystems, Inc. Printed in U.S.A.

Unlimited copying without fee is permitted provided that the copies are not made nor distributed for direct commercial advantage, and credit to the source is given. Otherwise, no part of this work covered by copyright hereon may be reproduced in any form or by any means graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an information retrieval system, without the prior written permission of the copyright owner.

TRADEMARKS

Sun, Sun Microsystems, and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd. All SPARC trademarks, including the SCD Compliant Logo, are trademarks or registered trademarks of SPARC International, Inc. SPARCstation, SPARCserver, SPARCengine, SPARCworks, and SPARCcompiler are licensed exclusively to Sun Microsystems, Inc. UltraSPARC is a trademark of SPARC International, Inc. AnswerBook, SunExpress, SunSolve CD-ROM, and SunSolve Online are trademarks of Sun Microsystems, Inc. Netscape is a trademark of Netscape Communications Corporation. All other product names mentioned herein are the trademarks of their respective owners.

For information regarding the SML Technical Report Series, contact Jeanie Treichel, Editor-in-Chief <jeanie.treichel@eng.sun.com>. For distribution issues, contact Amy Tashbook Hall, Assistant Editor <amy.hall@eng.sun.com>.

Contents

Preface.....	v
Acknowledgments.....	v
Credits.....	v
1 Introduction.....	1
1.1 Problems with classical indexing.....	2
1.2 Searching for information—an illustration.....	3
1.3 Synonymy and generality.....	4
2 Conceptual Indexing—a Better Way to Index.....	6
2.1 A fragment of a conceptual index.....	7
2.2 Using a conceptual taxonomy.....	9
2.3 Building a conceptual indexing and retrieval system.....	10
3 Elements of Conceptual Indexing.....	12
3.1 Concept extraction.....	13
3.1.1 Tokenizing.....	13
3.1.2 Markup and implicit markup.....	14
3.1.3 Lexical analysis.....	15
3.1.4 Morphological analysis.....	17
3.1.4.1 Inflection.....	18
3.1.4.2 Prefixes and suffixes.....	19
3.1.4.3 Prefix-suffix interaction.....	20
3.1.4.4 Lexical compounding.....	20
3.1.5 Phrase extraction.....	21
3.1.5.1 Interaction with markup.....	22
3.2 Concept assimilation.....	23
3.2.1 Conceptual structure.....	25
3.2.1.1 Quantificational tags.....	26
3.2.1.2 Conceptual relations.....	27
3.2.2 Conceptual subsumption.....	28
3.2.2.1 Relational abstraction.....	31
3.2.3 Organizing the taxonomy.....	32
4 Navigation and Retrieval.....	34
4.1 Dynamic passage retrieval.....	35
4.1.1 The relaxation ranking retrieval algorithm.....	36
5 Experience and Evaluation.....	39
5.1 Illustrative examples.....	39
5.1.1 Conceptual clustering.....	40
5.1.2 The power of subsumption axioms.....	42
5.1.3 Navigating in conceptual space.....	42
5.2 Formal measurements.....	45
5.3 The user experience.....	49
5.3.1 Indexed news articles.....	49
5.3.2 Indexed audio and video material.....	50
5.3.3 Online documentation.....	51
5.3.3.1 Efficient content-based access.....	53
5.3.4 Tools for working authors.....	55
6 Commentary.....	57
6.1 Penalty-based scoring.....	57

6.2 Effective search strategies.....	58
6.2.1 Refining a search.....	60
6.3 Cascaded indexing and retrieval	60
6.4 Distributed indexing and retrieval	62
6.5 Size of indexes	62
6.6 Semantic ambiguity	63
6.7 Comparison with related work.....	67
6.7.1 Related work in knowledge representation.....	67
6.7.2 Related work in information retrieval.....	68
6.7.3 Related work in morphology	69
6.8 Alternative approaches.....	70
6.8.1 Morphological variation.....	70
6.8.2 Semantic distance.....	71
6.9 Future directions	71
6.9.1 Publishing conceptually indexed material	72
6.9.2 Taxonomic documentation.....	73
6.9.3 Conceptually mediated hypertext	73
6.9.4 Conceptually assisted Web browsing	74
6.9.5 Other applications	74
7 Conclusions.....	75
8 References.....	79
9 Glossary	81
10 About the Author	91

Preface

The basic ideas leading to the technology of conceptual indexing go back to my earliest frustrations with the capabilities of "traditional" information retrieval technology, starting with an early version of Salton's SMART system, which I experienced as a graduate student in Salton's course on information retrieval when he was teaching at Harvard. These frustrations continued when I encountered some of the problems with automatic indexing in the context of a project with the Defense Documentation Center, using a system for automatic index-term extraction that they had developed. This indexing technique was included in the LUNAR question answering system that I developed, together with Ron Kaplan and Bonnie Webber (among others), at Bolt Beranek and Newman Inc. (BBN) for the NASA Manned Spacecraft Center in Houston, and it revealed some interesting limitations. These early frustrations were the motivation leading to the challenge I proposed to Ron Brachman, then my thesis student at Harvard, namely: how to describe the content of an individual sentence or phrase in such a way that it could be located in response to a specific description of need. This led to Ron's thesis on structured inheritance networks, which in turn led to the knowledge representation system KL-One, developed at BBN, where I introduced the notion of a most specific subsumer (MSS) algorithm, and to a subsequent international interest in "subsumption logic" or "terminological logics." In the research presented here, I return to the original idea of indexing material at the sentence and phrase level to support improved information access.

Acknowledgments

The research presented here was begun while the author was on leave of absence from ON Technology, Inc. and spending a year's "sabbatical" at Harvard University with support from the Kapor Family Foundation. It draws on research in taxonomic knowledge representation and natural language processing carried out at Bolt Beranek and Newman Inc. (BBN) with support from NASA, ONR and ARPA (alias DARPA, alias ARPA :-). The full implementation of these ideas and subsequent experimentation with this technology has been carried out over the past six years at Sun Microsystems Laboratories. The support of all of these institutions is gratefully acknowledged.

Credits

The development of this technology required the skills and cooperation of many people. I would like to give specific credit to the following people who participated in making this happen:

Firstly, to Jacek Ambroziak, who over the past six years has devoted heart and soul to bringing this vision to reality—implementing successive versions of the taxonomic classification algorithms and the relaxation ranking retrieval algorithm, developing the ConceptStore module that constitutes the core of the system, thinking deeply and making major innovations, and more than anyone else, making this vision his own and striving to design and implement whatever was required to make the system work.

Secondly, to Chris Colby, a summer intern from CMU, for implementing the first efficient MSS algorithm for this system.

To Peter Norvig, for creating our Lisp environment, for putting together the Recall system used to benchmark performance against traditional document retrieval technology, and for many other elements of our early explorations.

To Ellen Hays, for commentary and criticism of the early morphological rules and lexicon and for carrying out experiments with traditional document retrieval technology against which Recall was subsequently benchmarked.

To Larry Bookman, a postdoctoral researcher, for carrying out a series of experiments, including the analysis of performance of the Recall system that first demonstrated the degree to which this technology could improve the effectiveness of information access, compared to that of traditional document retrieval techniques.

To Tony Passera, for implementing an experimental graphical browser for conceptual taxonomies and implementing a rule-driven tokenizer module.

To Mark Torrance, a summer intern from MIT and subsequent contractor, for creating the first Mosaic interface to this technology, for playing a key role in developing the network-based client/server implementation, and for his enthusiasm and catalyzing effect on the group.

To Philip Resnik, for designing and carrying out the first user studies of this technology, for shepherding its initial incorporation into the Indexed Media Streams application, and for implementing the first phrase extraction module for the Nova implementation.

To Gary Adams, for implementing the morphological analysis component for the Nova implementation, for bringing the perspective of an experienced product developer to this project, for liaison to various groups within Sun, and for countless contributions to the functioning of the group.

To Patrick Martin, for taking on the deployment and technology transfer responsibilities for the Solar and Nova systems, for indexing a variety of materials and making them available to users over Sun's intranet, for thinking proactively about how to get this technology in the hands of real users, and for diligently taking care of the countless chores that it takes to make a technology real.

To Robert Kuhns, a consultant and contractor, for developing our English grammar, for surveying the capabilities of commercial information retrieval systems, for advice and guidance to many members of the group, and for checking out many lexical entries generated by automatic morphological analysis to determine which ones were erroneous and what the correct analysis should be.

To Cookie Callahan, our group administrator, for helping to check lexical entries, and for supporting the group in uncountable ways.

To Bob Sproull for his patience, insights, and advice, and for his assistance in laying out the architecture for Textbase and implementing its first markup analyzer.

To Wayne Rosing and Bert Sutherland, my successive lab directors at Sun, for their faith, encouragement, and support.

To Mitch Kapur, for his inspiration to build systems that are easy to use and truly meet the needs of their users and for his support and encouragement to get this started.

To James Fallows, for "kicking the tires" of this technology, for the excitement generated (his and ours) when he found it valuable for his own information needs, and for writing about the experience in the *Atlantic Monthly*. Also for his permission to use a collection of his articles and commentaries in a demonstration system.

And finally, to my wife Selina, a high school teacher and library media specialist (aka librarian), whose experiences looking for things for her students using online search services and Internet search tools have sharpened my intuitions about the needs of people trying to find information, and my two sons Matthew and Alexander, who together with Selina put up with a husband and father who spent many late nights, weekends, and holidays hacking lexicons and morphological rules, debugging phrase extraction and concept assimilation algorithms, working late with members of the project team, getting out of bed in the middle of the night to write down ideas, and generally displaying a fascination with esoteric facts about dictionaries and word meanings that has been mystifying to behold.

Conceptual Indexing: A Better Way to Organize Knowledge

William A. Woods

Sun Microsystems Laboratories
2550 Garcia Avenue
Mountain View, CA 94043

1 Introduction

How often have you tried to look up something in an index and failed to find what you were looking for because the words or phrases you looked for were different from those used in the material you needed to find? For many people, this is a familiar experience in such diverse contexts as looking for something in a telephone directory, searching the index of a product catalog, consulting the index in the back of a book, or searching the topics list of an online information service. Many of the same problems arise when trying to use a text retrieval system based on keyword or full-text indexing to search a library collection or an index to pages on the World Wide Web. This paper presents a new technique for organizing information to support subsequent access that can dramatically improve your ability to find the information you need, with less hassle and with better results.

Conceptual indexing combines techniques from knowledge representation and natural language processing with classical techniques for indexing words and phrases in text to enable a retrieval system to make connections between the terminology of your request and related terminology in the information that you need. A conceptual indexing and retrieval system can dramatically reduce the frustrating struggle you have probably experienced trying to discover the "magic words" that will connect you to the information you seek. It does this by automatically analyzing the conceptual structure of phrases extracted from the material, and using semantic relationships between words and concepts to establish connections between the terminology used in your request and other related terminology that may provide the information you need. It automatically organizes all of the words and phrases of the indexed material into a "conceptual taxonomy" that explicitly links each concept to its "most specific subsumers" (i.e., the most specific known concepts that are more general than the concept in question) and to other semantically-related concepts (e.g., those in which this concept is a modifier). This structure is then used to find connections between terms of a request and related concepts in the index.

The conceptual taxonomy, organized by the most-specific-subsumer (MSS) relationship, provides a topological structure for the space of conceptual descriptions that can be navigated to find concepts related to each other, and can also support efficient conceptual search and retrieval. In such a taxonomy, specialized algorithms can efficiently find all of the subsumers of a concept (i.e., all of the concepts that are more general than that concept) and all of the subsumees of a concept (i.e., all of the concepts that are more specific). Searching for connections in this structure between terms of a request and other known concepts provides a partial solution to the "paraphrase problem"—the situation that arises when the terminology used in a request does not match exactly with that used in the information sought.

1.1 Problems with classical indexing

The classical technology for organizing information in libraries and encyclopedias and in the indexes at the backs of books is based on manually organizing topics into fixed hierarchies and/or listing topic phrases in alphabetical order. Although these are standard and universal techniques, they have serious limitations when it comes to supporting the needs of an information seeker.

Hierarchical topic trees are useful for outlines and tables of contents of books, but they start to break down when applied to whole library collections or to topic lists in online information services when they contain more than a few levels of depth or more than a few thousand topics. Navigating through such a hierarchy can begin to feel like trying to find your way through a maze, when the size of the hierarchy becomes large. This is because at any given level in such a hierarchy there may be more than one choice of general topic under which the information sought might be classified, so each possibility may have to be searched separately. If each of two or more choices at one level lead to two or more choices at a later level, and so on for more than a few levels, then the total number of places to look can become cumbersome large and the difficulty of remembering the places that have not yet been checked becomes unmanageable.

Hierarchical library classification systems (such as the Dewey decimal classification system or the Library of Congress (LC) classification system) are attempts to develop static, hierarchical classification structures into which all of human knowledge can be classified. However, strict hierarchical systems like Dewey and LC are unable to fully capture all of the desired relationships because of the necessity to place each concept in exactly one place in the hierarchy, a limitation that arises partly as an artifact of the use of prefixes of the category codes to express the generality relationship. This means that, for example, one cannot put "cleaning automobiles" (say) directly under both "cleaning" and "business opportunities," because if these latter two concepts had been assigned the codes (say) 37 and 98, then one would have to choose only one of them to use as the basis for assigning the code (say) 37.8 to "cleaning automobiles."

As a result of this single-parent restriction, a cataloger often has to choose arbitrarily among several reasonable categories to assign as the catalog code for a new document. Despite use of documents called "subject authorities," which attempt to impose some controls on terminology and classification criteria, there is no guarantee that two catalogers would make the same decisions. An information seeker therefore has to second guess the catalogers to decide where to look for an item and will typically have to look in a number of places to find what is needed (assuming he or she is clever enough to think of all the relevant plac-

es to look). Another limitation of these classification systems is that they are static and do not evolve fast enough to adequately track the changing terminology of evolving fields of published knowledge.

Likewise, simple alphabetical ordering of lists of topics has serious limitations because it is often difficult to guess the exact sequence of words used to describe a topic. Guessing incorrectly can lead to looking for the desired information in the wrong place. For example, anyone looking for "automobile brokers" in the 1996 NYNEX Yellow Pages for the Boston Area (if they don't know that this has been indexed as "Automobile & Truck Brokers") will probably stop looking when nothing is found between "Automobile Brake Business" and "Automobile Bumpers." It would never occur to most people to look for the desired entry 61 pages later, after "Automobile Transporters & Drive-Away Companies."

The problem of guessing where to look is often complicated by the very techniques some catalogers use to try to improve the situation. For example, if you were looking for "industrial steam cleaning" in the above mentioned Yellow Pages, you might eventually find the entry for "Steam Cleaning-Industrial," but probably after failing to find anything useful between "Industrial Security" and "Infant's Wear," and probably after following a fruitless cross-reference from "Industrial Cleaning" to "Cleaning Svce.-Industrial," and then only if you are persistent, are good at problem solving, have some knowledge of the way index entries are often constructed, and either have good luck or good intuitions.

The problem is worse when the words you are looking for are different from those of the item you need. For example, if you're looking for "automobile cleaning" when the desired information is indexed under "Car Washing," then alphabetical order doesn't help much. When variations in word choice as well as word order must be taken into account, the space of possibilities from which to guess alternative phrasings can become dauntingly large. Moreover, even when a relevant entry has been found, there is often no guarantee that better information may not be indexed under some other heading that has not yet been searched. For example, in one Yellow Pages directory that I examined, there were three different headings dealing with rustproofing of cars: "Automobile Undercoating & Rustproofing," "Rustproofing," and "Rustproofing & Undercoating-Automotive." These three headings had different listings and were only partially cross-referenced. If you found one of them, you might think you had found everything, but you would be wrong.

1.2 Searching for information—an illustration

The problems with classical indexing techniques can be illustrated with a real example of a search I made in my local Yellow Pages. In need of a source for automobile steam cleaning, I found nothing useful when I first looked for "steam cleaning," which didn't occur directly as a heading, but occurred in the phrases "Steam Cleaning Equipment" and "Steam Cleaning-Industrial." When I then looked for "automobile steam cleaning," and subsequently for "automobile cleaning," I found nothing useful, although there was an entry for "Automobile Cleaning Equipment." The most useful heading, it turns out, was "Car Washing & Polishing," which was cross-referenced from an entry "Automobile Washing & Polishing." The cross-reference was clearly important here, since this is an exception to the rule (which I had assumed was universal) that entries about cars are indexed under "Automobile."

Unfortunately, the fact that the cross reference was listed in alphabetical order under "Au-

tomobile Washing & Polishing," rather than "automobile cleaning," means that it came at the end of the automobile section rather than the beginning. Finding it required searching the entire automobile section. This consisted of many pages in the Yellow Pages I was searching (which had no index in the back) and took up almost a whole page of entries in the index in the back of another Yellow Pages directory I checked later that had an index. Disappointingly, even after all of this searching, the fact that "Car Washing & Polishing" was the best entry for finding information about "automobile steam cleaning" was not explicitly indicated in the Yellow Pages, and can't be deduced without additional knowledge. I had to call a car wash to find out.

These particular problems may be unique to my local Yellow Pages, but they are typical of problems that occur in all indexes, whether in the back of a book, in the catalog of a library, or the directory of an online information service. They are also present in manually-linked hypertext or hypermedia documents and in any full-text search and retrieval system. They stem in part from the fluency and subtlety of natural language and from the virtual impossibility for a human indexer to notice, catalog, and cross-reference all possible variations in phrasing. Moreover, when an index is printed on paper, as with a printed directory or an index in the back of a book, recording all such cross-references would substantially increase the size of the index. Consequently, a printed index will always be a compromise—attempting to provide enough information to be useful, but filled with residual problems that can make using the index a tedious problem-solving activity. Fortunately for online computerized indexes, this aspect of size is not a problem, and systems for online indexing and search can do substantially better.

1.3 Synonymy and generality

The reaction of many people who are familiar with information retrieval techniques, when confronted with the problems described above, is to propose the use of a synonym thesaurus to expand a request by adding terms that are synonyms of the terms requested. Such a thesaurus is sometimes used by full-text search and retrieval systems, most of which employ search algorithms that ignore word order and simply count the number of terms of the (perhaps expanded) query that occur in the target text. Some such systems allow for different weighting of different terms rather than simply counting them (in order to capture the fact that some terms of a request may be more important than others). Unfortunately, attempts to automatically expand queries using a synonym thesaurus often fail to improve the retrieval effectiveness of these systems and frequently actually degrade performance. Part of the reason is that there are very few true synonyms in English (or any other language for that matter), and members of the "synonym" sets in such thesauri often differ significantly in meaning. Typically there is a difference in generality, as in the set: {automobile, car, truck, bus, taxi, motor vehicle}, where the last is clearly much more general than any of the others, and in this case effectively summarizes the others.

Choosing to treat terms as if they were synonyms, when they really aren't fully synonymous, introduces a level of granularity in the retrieval process that trades off precision against recall. Treating such terms as synonymous amounts to generalizing the query to the level of abstraction at which the differences among the individual "synonyms" does not matter. Unfortunately, there is no a priori level of generality that is correct for all information needs. For example, if you were to ask for "motor vehicle" then you would probably expect to pick up hits for "car," "truck," and "bus," but if you asked for "automobile" then

you would presumably not want to get "truck" and "bus." If requests are automatically expanded by the addition of terms drawn from a fixed synonym thesaurus, then for any large number of queries, it is often the case that the queries whose recall results are improved by the query expansion are offset by other queries in which the expansion produces unwanted noise that dilutes the precision of the result. In a system where the results are ordered and at most a specified maximum number of items are retrieved, this can result in a decrease in recall as well as precision, because additional "noise" items can displace correct items.

One can improve this situation by matching requests to targets using a relationship of generality rather than synonymy. In this case, a term in a request will match any corresponding term in a target that is at least as specific as the requested term. Thus, a request for "motor vehicle" would retrieve all kinds of motor vehicles, while a request for "automobile" would retrieve cars and taxis but not trucks and buses. We can express this notion of generality as a relationship of *conceptual subsumption*, where a more general term is said to "subsume" a more specific term. Formally, a term also subsumes itself and subsumes any true synonyms that it may have. Thus, subsumption is more general than synonymy (i.e., subsumption subsumes synonymy). True synonymy is equivalent to mutual subsumption. If a retrieval system is designed to retrieve all items that are subsumed by a request, then the information seeker has a way of controlling the level of generality of the search by choosing the level of generality of the query terms, thus avoiding a major source of precision/recall trade-off.

Some information retrieval systems make use of a hierarchical thesaurus rather than a simple synonym thesaurus, although nothing constrains them to using only subsumption as the basis for the hierarchical organization. (Other such relationships might include: part-whole relationships, geographical or political subdivisions, hierarchical organizational structure, and arbitrary topic-subtopic relationships such as that between cameras and film). Nevertheless, subsumption is typically a dominant relationship in such hierarchical thesauri, and such systems can partially address the limitations of synonym thesauri discussed above (by using only hierarchically-dominated topics to expand a query). Even in these systems, however, automatically expanding requests often fails to increase retrieval effectiveness and may actually degrade it. Accordingly, most commercial systems that include hierarchical thesauri provide them as an adjunct that an information seeker can refer to, but do not automatically expand a user's request.

Part of the reason that hierarchical thesauri often don't help is that the topics in these thesauri are "atomic." That is, even when a topic description involves multiple words, there is no exploitation of the structure of the phrase and how it might relate to other paraphrases or other related phrases (except for the explicit hierarchical relationships that have been anticipated and carefully crafted by the human indexers who constructed the hierarchy).

There is no way in such systems to automatically relate a phrase in the hierarchy to a new variation or related phrase that occurs somewhere in an item of material being indexed. Thus, the only way that a phrasal topic in such systems can be related to an indexed item is when that phrase occurs literally in the indexed item or is assigned to that item by a human cataloger. Phrasal topics in these systems function as if they were simply large words. Unfortunately, because of the statistics of word matching, the likelihood of an exact phrasal match is often small. (The exception is when the phrase is a name or a common "frozen" expression that really does function as if it were a single word.)

An exception to the above generalization about hierarchical thesauri in commercial systems

is the Topic system from Verity, although strictly speaking, the "topic" hierarchies in this system are much more complex and powerful than simple hierarchical thesauri. In Topic, each level of hierarchical expansion is a manually-constructed search specification involving weighted combinations of a variety of query operators. This provides a powerful ability to tune the performance of an information request, but designing these "topic" specifications is a labor-intensive activity that requires special skill. It involves a good deal of problem solving and good intuitions about the relationships between words and the way words are used in the material being indexed.

Hierarchical thesauri (and the "topic" hierarchies of systems like Topic) are manually created structures that require considerable skilled manual labor to construct. The location of each concept in the hierarchy is determined by a human decision, and only the relationships between topics that have been explicitly identified and added by a human designer are represented in these thesauri. In the next section, we will describe a different approach that goes beyond the capabilities of a manually-constructed hierarchical thesaurus. This new approach, based on an automatic analysis of phrasal concepts and an understanding and exploitation of their conceptual structure, improves the functional utility of the resulting index and dramatically reduces the human labor required.

2 Conceptual Indexing—a Better Way to Index

Conceptual indexing is a methodology that allows indexing systems to take advantage of the conceptual structure of phrases in the indexed material. It does this by automatically parsing each phrase into one or more conceptual structures that represent how the elements of the phrase are assembled to construct its meaning(s). This allows a system to automatically determine when the meaning of one phrase is more general than that of another, given what it knows about the generality relationships among the individual elements that make up the phrase. For example, a system can automatically determine that car washing is a kind of automobile cleaning if it has the information that a car is a kind of automobile and that washing is a kind of cleaning. Thus, a system can automatically test for generality relationships between structured concepts if it has a foundation of basic "known" facts expressing subsumption relationships between basic or "atomic" concepts. We refer to these basic known subsumption relationships as "subsumption axioms." These basic axioms are originally created by human lexicographers or "knowledge engineers" (or in some cases by morphological rules), and are represented in a lexicon or an initially-constructed taxonomy from which they can be accessed by the subsumption algorithms as needed.

The creation of a knowledge base of basic subsumption axioms is a substantial undertaking, and the lack of such basic knowledge has been a limiting factor in many natural language processing applications. This project has created a knowledge base of approximately 30,000 subsumption axioms to support the taxonomic organization of English words and phrases described above. Another source of such facts, which we have also used, is George Miller's WordNet [Miller, 1995], which records basic relationships among more than 60,000 words. Although both these knowledge bases include nouns, verbs, adjectives, and adverbs, the WordNet taxonomy has a more complete coverage of nouns, whereas our axioms have given more attention to taxonomies of verbs. (A related, but different, large-scale knowledge-base effort is Doug Lenat's Cyc project [Lenat and Guha, 1990], which focuses on nontaxonomic, general world knowledge.) Once a base taxonomy of lexical

subsumption axioms has been developed, this can be used to automatically determine the relationships among a much larger class of structured phrases.

The ability to formally decide when one phrase is more general than another enables a system to automatically organize all of the words and phrases extracted from indexed material into a conceptual taxonomy in which each word and phrase is explicitly linked to its most specific subsumers (MSSs)—i.e., the most specific concepts that are more general than it is. The taxonomy can also be expanded to include additional words and phrases, such as related terms extracted from a lexicon or terms that are input by information seekers expressing their interests. A conceptual taxonomy constructed according to this methodology enables search and retrieval algorithms to find connections between terms used in a request and more general and more specific terms that may be of interest. The taxonomy can also be used as a "conceptual map" of the content of the material and can be used by an information seeker to navigate the material in "conceptual space."

A key attribute of this technology is the ability to determine where in the taxonomy a new description would belong, when the description itself is not already in the taxonomy. That is, an algorithm called the MSS algorithm can efficiently find the most specific subsumers of a new description, even when that description itself does not occur in the taxonomy. This allows information seekers to find concepts related to their requests, even when the requested terminology is not explicitly in the taxonomy. This is an improvement over the abilities of a static, manually-constructed thesaurus, because it allows an information seeker to have success even when the phrasing of a request has not been anticipated and is not in the taxonomy.

There has been considerable research on taxonomic representations of knowledge within the artificial intelligence research community (see, e.g., Woods & Schmolze [1992]), much of it based on the notion of *subsumption* and other issues discussed here. The conceptual indexing project draws results and insights from this body of research and extends it in some interesting ways. In order to keep this presentation focused, I will postpone discussion of this background state of the art and the relationship of this work to other work until the commentary section near the end. To maximize the accessibility of the paper to those who may not be familiar with some of the different areas of research or background knowledge involved, I will attempt to motivate the methodology from first principles and avoid nonintuitive technical jargon wherever possible. To help with the necessary technical terms from various disciplines, as well as the new terminology introduced here, a Glossary is included at the end of the paper.

2.1 A fragment of a conceptual index

At this point, an example is in order. A portion of a conceptual taxonomy for some of the phrases involved in our previous discussion of Yellow Pages searching is shown in Figure 1. The phrases in this taxonomy (with the exception of the two occurrences of the request "automobile steam cleaning" marked by exclamation marks) were either extracted from the Yellow Pages or derived from such phrases by dropping modifiers to produce more general descriptions. The structure of the taxonomy was derived automatically by a *classification algorithm* that is a combination of the MSS algorithm mentioned above and a related MGS algorithm that finds most general subsumees. The two occurrences of "automobile steam cleaning" illustrate where this phrase would be located in the taxonomy by the classifica-

tion algorithm if it were entered as a request but did not already exist in the taxonomy.

Notice that some of the phrases occur multiple times and have been numbered in order of occurrence (in parentheses following the phrase). These represent nodes that have multiple parents (i.e., multiple most specific subsumers) in the taxonomy. Unlike the hierarchical relation in the Dewey system or in the Library of Congress classification system, taxonomic subsumption allows a concept to have multiple "parent" concepts that directly subsume it. Thus, for example, a conceptual indexing system can (and will) put "automobile upholstery cleaning" directly under both "automobile cleaning" and "upholstery cleaning," so an information seeker can expect to find it under either topic without having to guess which is most likely.

```
brokers
  automobile brokers
  truck brokers
cleaning
  automobile cleaning
!     automobile steam cleaning (1)
      automobile upholstery cleaning (1)
      automobile washing (1)
      car washing (1)
  industrial cleaning
      industrial steam cleaning (1)
  steam cleaning
!     automobile steam cleaning (2)
      industrial steam cleaning (2)
  upholstery cleaning
      automobile upholstery cleaning (2)
  washing
      automobile washing (2)
      car washing (2)
equipment
  automobile equipment
      automobile cleaning equipment (1)
  cleaning equipment
      automobile cleaning equipment (2)
      steam cleaning equipment (1)
  steam equipment
      steam cleaning equipment (2)
```

Figure 1. A fragment of a conceptual taxonomy.

Legend: Exclamation marks indicate location of a new request phrase that is merely located in the taxonomy but not part of it; numbers indicate successive occurrences of concepts with multiple parents.

Although it appears otherwise in the figure, there is actually only one copy of any given concept in a conceptual taxonomy, even when the concept has multiple parents. Such concepts are printed out separately under each parent in this outline presentation for readability. There are various ways of displaying hierarchical structures in which nodes may have multiple parents; this outline form is only one of them.

Remember that all of the phrases in the taxonomy were mechanically derived from phrases that occurred in the Yellow Pages listings, with the exception of the two occurrences of the query (marked by exclamation marks). Also remember that the location of the query phrase within this taxonomy (indicated by exclamation marks) was automatically determined by the MSS algorithm.

2.2 Using a conceptual taxonomy

If the information in our Yellow Pages search example had been organized using the above conceptual taxonomy, and if an MSS algorithm had been available for finding the most specific subsumers in the taxonomy for a new request description, then our "automobile steam cleaning" request could have immediately produced the information that the taxonomy contained entries for "automobile cleaning" and "steam cleaning" but not "automobile steam cleaning." This information could be used to present a display of relevant fragments of the conceptual taxonomy showing these phrases and their subsumees. Such a display might look like that in Figure 2.

```
automobile cleaning
    automobile upholstery cleaning +
    automobile washing -
    car washing +
steam cleaning
    industrial steam cleaning +
```

Figure 2. A topical view from a conceptual taxonomy.

The entries marked with plus signs in this display are descriptions under which there are listings. Those marked with a minus have only a cross reference to another title or titles. The other (unmarked) phrases are automatically derived phrases used to organize the taxonomy.

Notice that using the conceptual taxonomy, an information access system could quickly produce a concise summary of everything in the Yellow Pages that might relate to the request "automobile steam cleaning." A person would then need only a glance to rule out upholstery cleaning and industrial steam cleaning and realize that car washing is the best bet. This replaces what for me was a substantial amount of page turning and scanning in

order to glean the corresponding information from the alphabetical listing of my physical Yellow Pages.

2.3 Building a conceptual indexing and retrieval system

In addition to applications like the Yellow Pages (where the phrases to be indexed are directly provided by the application), conceptual indexing can also be applied to full text or to the contents of databases by automatically extracting words and phrases from the source material and remembering the positions in the indexed material where the terms were found. These words and phrases can be automatically organized into a conceptual taxonomy that will serve as a deep "conceptual map" of the source material and can enable an information seeker to navigate within the material using the conceptual taxonomy as a guide. The conceptual taxonomy can also help text search algorithms deal with the paraphrase problem by making connections between words and phrases used in a request and related terms that may be found in relevant material. Not only can conceptual indexes support traditional *document retrieval*, they can also support a new kind of *passage retrieval* that uses the index information about where concepts occur in the indexed material to identify and rank specific passages that may contain the specific information requested. We call this "*dynamic passage retrieval*." Such systems can provide a user with much more effective access to information by minimizing the reading time required to discover where (or whether) the information sought can be found within the retrieved documents.

In order to build a conceptual indexing and retrieval system to support the kind of information access described above, it is necessary to solve a number of problems. First is the need for an application to provide specific phrases to be indexed or to have a procedure for extracting such phrases automatically from the material to be indexed. Next is the need to transform these natural language phrases into internal representations that will capture their conceptual structures and support the operations of indexing, searching, and matching. Then it is necessary to organize these internal descriptions into a structure that can be efficiently searched, and it is necessary to have efficient algorithms that can search this structure to find the most specific subsumers of a specified request concept. It is also desirable to have tractable algorithms for organizing large collections of conceptual descriptions into such a structure. To support identification of relevant passages in source material, it is necessary to have an efficient search algorithm that can use the conceptual index to make connections between terms in a request and related terms that may occur in relevant passages, and to locate such passages in the text material. Finally, it is necessary to have a user interface that can display portions of the taxonomy, allow a user to navigate around in the taxonomy, and to shift attention back and forth between the conceptual taxonomy and passages of indexed material where concepts occur.

There is a great deal of work required to build up the infrastructure necessary to extract concepts from text and organize them into a conceptual taxonomy and then to browse this taxonomy and use it to locate specific passages in text. This infrastructure includes components for:

document handling:

- a tokenizer that segments a sequence of characters into a sequence of words and symbols, punctuation marks, and document structure indicators.

- a markup interpreter that recognizes and allows the system to deal with systems of explicit markup such as SGML and HTML and several older markup conventions such as Troff and Scribe.
- an *implicit markup* analyzer that deals with the structure implicit in ordinary text which has not been marked up but which has been organized to be readable by a human.

lexical analysis:

- a lexicon and knowledge base of syntactic and semantic information about words and idiomatic phrases, including information about different senses of words and different interpretations of phrases. The lexicon (or a collateral knowledge base) also contains *subsumption axioms* that express generality relationships between concepts associated with known words and phrases and their senses. These subsumption axioms are used to ground out the process that tests for generality relationships between structured phrases.
- a collection of morphological rules for recognizing regularly-inflected forms of words, for recognizing words derived from other words by the addition of prefixes and suffixes, for recognizing words formed as compounds of known words (e.g., shoelace), and for guessing syntactic and semantic information about unknown words.

phrase analysis:

- a phrase extractor for extracting phrases from text and selecting those phrases to be indexed.
- a parser and grammar for recognizing the structure of phrases and transforming them into conceptual structures that can be assimilated into the conceptual taxonomy.

taxonomic classification:

- a concept assimilator that adds words and phrases into the conceptual taxonomy and determines any other related concepts and conceptual relationships that should be added (such as more general concepts derived by dropping modifiers, more general concepts found by consulting subsumption axioms or morphological relationships, and any concepts and relationships derived from special rules or explicitly indicated in a lexicon).
- the MSS and MGS (most general subsumee) classification algorithms necessary to efficiently place new concepts at the correct place in an evolving taxonomy—i.e., under the appropriate most specific subsumers and above the appropriate most general subsumees.

browsing and retrieval:

- a search and retrieval algorithm that can use the conceptual taxonomy to make connections between terminology used in a request and other related terminology that may be used in relevant material and that can locate specific passages of text where answers to a request are likely to be found.
- an interactive presentation/navigation system suitable for displaying portions of the taxonomy and allowing an information seeker to effectively move around in the taxonomy in search of information and to move conveniently back and forth between the conceptual taxonomy and locations in the indexed material.

All of this machinery has been implemented in a series of exploratory conceptual indexing and retrieval systems developed at Sun Microsystems Laboratories in Chelmsford, Massachusetts over a period of six years. Successive systems, named respectively, *Pilot*, *Recall*, *Solar*, and *Nova*, have been used to determine the feasibility and utility of these ideas and to develop initial versions of the necessary lexicons, rules, algorithms, and knowledge bases. *Nova 1.0* is a version of this technology that has been implemented in C and C++ as a deployable system that can be incorporated into a variety of search and retrieval applications. The *Nova* system is constructed using *Textbase*, a suite of natural language processing and phrase extraction modules, and *ConceptStore*, a general purpose knowledge representation system that supports conceptual taxonomies and lexicons and text indexes, and provides the algorithms for taxonomic classification and dynamic passage retrieval.

The *Pilot* system was designed to develop and explore the methodology of conceptual indexing. This system continues to be used to explore new kinds of material and to develop new capabilities. The *Pilot* system is the crucible in which the basic lexicons, rules, and algorithms are forged. The *Recall* system was designed and used to experiment with the basic ideas of dynamic passage retrieval and to compare the capabilities of this approach to those of traditional document retrieval systems. The *Solar* system is a prototype network-based server that was implemented and deployed to support experimentation with users of this technology, indexing a variety of material for various information needs. *Solar* has been incorporated, for example, into a version of Sun's Indexed Media Streams application [Northcutt, 1995] to index closed-captioned audio visual material. Both *Solar* and *Nova* have been used to index a variety of material relating to Sun's Java(TM) language and various bodies of system support documentation.

All of these systems automatically extract phrases from the text to be indexed (usually simple noun phrases, although some also extract individual adjectives and verbs and simple verb phrases). These phrases are then assimilated into a conceptual taxonomy that also records where the phrases occurred in the indexed material. (The term "indexed material" is used here to cover various kinds of indexed items, including traditional documents, Internet Web pages, e-mail messages and archives, descriptions of items in product catalogs, document titles, document section headings, bug report descriptions, unrestricted text files, and closed-captioned audio visual streams.) The resulting conceptual taxonomies are then used to support a dynamic passage retrieval algorithm that finds, scores, and ranks specific passages in the material that it concludes are likely to contain information relevant to the request. The objective of these systems is to be helpfully responsive to a spontaneous description of the information required, minimizing the need for an information seeker to engage in repeated query reformulation in order to discover the exact terminology that will retrieve the information required.

In the next two sections, I will briefly address the approach of these systems to each of the above problem areas, first addressing the construction of the conceptual index, and then its use for navigation and retrieval.

3 Elements of Conceptual Indexing

The famous architect Mies van der Rohe once said that "God is in the details." I have found this slogan to be an appropriate guiding principle for advancing the state of the art of information access. In this section, I will attempt to give an impression of the many details that

have been addressed in arriving at the current methodology and some of the solutions that collectively add up to a significant improvement in the ability for people to find information. Although I will attempt to be brief in dealing with each issue, the total number of such issues is fairly daunting. If you want to read less, I suggest that you skip this section for now and come back to it later for more detailed information. You may want to continue with the next major section, Navigation and Retrieval, which describes how this technology can be used for improved information access.

In the rest of this section, I will first discuss techniques for extracting and analyzing concepts from texts. I will then address the conceptual structures of natural language descriptions that determine how they should be placed in a taxonomy and introduce some of the subtleties of the notions of subsumption and taxonomy as they relate to conceptual structure. I will briefly describe how a conceptual taxonomy is organized for efficient searching and how it is used for searching and navigation. Remember that a key feature of this technology is the ability (illustrated in the introduction) to find useful information even when the requested concepts do not occur in the taxonomy. The purpose of many of these details is to support that capability.

3.1 Concept extraction

At a high level, conceptual indexing technology can be thought of as a combination of (1) a *concept extractor* that identifies words and phrases to be indexed (and records where in the material they occurred); (2) a *concept assimilator* that analyzes the structure and meaning of a concept phrase to determine where in the conceptual taxonomy it belongs (and what other concepts it should be related to, which may themselves need to be added to the taxonomy); (3) a *conceptual retrieval system* that uses the conceptual taxonomy to make connections between requests and indexed items of information; and (4) a *conceptual navigator* that allows a user to browse the conceptual taxonomy and to move back and forth freely between concepts in the taxonomy and occurrences of those concepts in the indexed material. In this section I will describe the elements that go into making a concept extractor, focusing primarily on the Pilot system for the sake of concreteness. In subsequent sections, I will take up the topic of concept assimilation, navigation, and retrieval.

3.1.1 Tokenizing

In order to extract concepts from text, it is first necessary to interpret the text as a sequence of words (and other word-like items such as numbers, date strings, e-mail addresses, etc.), occasionally interrupted by punctuation marks and other orthographic markers (such as carriage returns and indentation). This is the job of the *tokenizer*, which segments a character stream into a sequence of *tokens*. Now you might think that tokenizing is a simple low-level capability that requires no real attention, and indeed most retrieval systems do not seem to pay much attention to it. However, the way a text is tokenized can make a big difference in whether certain searches will succeed. Moreover, the criteria for tokenizing interact in a delicate dance with the lexical analysis phase of processing and the ability to correctly identify phrase boundaries. If there is a discrepancy between how the tokenizer views a "word" and how the lexicon views it, then the two may fail to connect when they should. For example, if the tokenizer always breaks words at apostrophes, then there will never be a token "o'clock" or "O'Grady" to look up in the lexicon.

The goal of the tokenizer is to group characters into units that can be looked up in a lexicon to determine what is known about them and how to treat them. These units include not only ordinary English words, but also various combinations of alphabetic and special symbols such as date strings (2/21/95), words with internal apostrophes (o'clock), phone numbers (442-0435), e-mail addresses (william.woods@east.sun.com), hyphenated words (drop-kick), etc. For the sake of simplicity, we will call all such units "word tokens." Thus, a word token consists of a sequence of characters and special symbols that will be used as a search key for lexical lookup.

An important forcing function for deciding what characters should be included in a token is the utility and efficiency of lexical lookup of items. This is because it is highly advantageous to tokenize a string of characters that has a coherent identity as a single token so that it can be looked up in a lexicon, analyzed once and added to a working lexicon (if not previously known), and then subsequently looked up each time it reoccurs in the material. This is preferable to dividing such a string into multiple elements that must be repeatedly parsed into a coherent whole each time they occur.

Thus, for example, in the Pilot system, hyphens and slashes are tokenized into words so that path names, ratios, date strings, etc. are tokenized as a single unit. On the other hand, apostrophes are broken off from the beginnings and endings of words because they are usually quotation operators or possessive markers that are not part of the lexical identity of the word. Internal apostrophes, on the other hand, in words like "o'clock" are kept. A special exception is made for "s" at the end of a word, which is broken off as a separate token, indicating plural or possession. Contractions using apostrophes can be dealt with lexically or by morphological analysis so they are not broken out of tokens. Similarly, words are not broken at internal periods or commas (e.g., for e-mail addresses, decimal and/or large numbers, etc.), while periods and commas that are followed by white space are broken off. This behavior is table-driven so that it can be changed for specialized situations, but considerable care has been devoted to defining a default behavior that will be correct in the widest possible variety of foreseeable circumstances. This is just a sampling of the details that go into making a good tokenizer. Other issues include context-sensitive tokenizing in order to deal with specialized tokenizing conventions for specialized contexts.

Notice that the tokenization criteria necessary for dealing with certain phenomena depend not just on a classification of some characters as break or separator characters but also on the context in which particular characters are used. When designing a table-driven tokenizer, most system designers would not be likely to include the capability to represent such rules if considerable thought weren't first given to the kind of tokenization behavior necessary for dealing with the nontypical cases and with the interaction of tokenization behavior with the rest of the system.

3.1.2 Markup and implicit markup

In designing the Pilot system, a great deal of care was given to the objective of being able to index unrestricted text without requiring a preprocessing stage to standardize the format of the material. This was done so that it would be easy to directly apply the system to many new bodies of material, and to evolve a system that was broadly effective across many domains. This means that the system may be given material to index, some of which may be marked-up source for some document formatting or display system and some of which may be ordinary ASCII text files already formatted for viewing as text. It is useful to know

when the material is markup source so as not to waste resources indexing the markup operators, but the most important reason to make this distinction is the effect markup has on phrase boundary decisions (to be discussed shortly). When the text is intended to be viewed directly, orthographic features such as short lines, indenting, and extra carriage returns determine an "implicit markup" that affects phrase boundary decisions (whereas such features are irrelevant in markup source files, where explicit markup determines phrase boundary decisions, and the line lengths in the source text are often left random when human beings edit these files because they know this will have no effect on the output).

Consequently, one of the first steps in the Pilot indexer, when it processes a file, is to examine the first 2500 characters of the file (an arbitrarily chosen number that seems to work) to determine whether the file appears to be already formatted text or a recognizable kind of markup source. In the already formatted case, it uses the implicit markup information indicated by lengths of lines, indenting, and carriage returns to determine when a phrase should be forced to end at the end of a line or should be allowed to flow onto the next line. In this case, it also looks at runs of white space and alignments of columns in successive lines to identify regions that appear to be tables so that it can block phrases from flowing from one table column into the next.

In the explicit markup case, the indexer uses the markup operators themselves to determine when phrases should be blocked from continuing. It does this by using knowledge of the escape characters and operators of known markup conventions to determine what markup convention is being used; it then uses table-driven information about the inferred markup convention to determine which markup operators allow phrases to flow through them (e.g., emphasis operators like italics), which ones force phrase boundaries (e.g., paragraph delimiters), which ones enclose information that should not be indexed (e.g., formatting operators that set document parameters such as page margins), and which ones enclose regions that should be indexed, but should be indexed out of sequence while allowing phrases to flow around them (e.g., footnote wrappers). The markup conventions currently known to the Pilot system are HTML and the basic operators of several common document formatting languages such as Scribe, Troff, and Docformat.

Because it takes this approach of automatically recognizing markup and implicit markup, the Pilot indexer can usually be applied to new material by simply giving it a path name to a directory where the material is stored, without any special preprocessing or formatting. This has made it possible to apply the system regularly to new kinds of material, and this in turn has contributed significantly to the robustness of the system. Of course, avoiding the need for specialized preformatting is also a great convenience to many future users of this technology.

3.1.3 Lexical analysis

Once a token has been identified in an input text, it can be looked up in a lexicon to determine what to do about it. How it is processed can depend on the answers to many questions that can be encoded as feature-value pairs in a lexical entry. For example, can the current word continue a current phrase?; can it start a new phrase?; should it block a phrase or terminate a phrase?; should it be indexed all by itself?; does it participate in any known idiomatic phrases?; is it a noun, verb, or adjective?; is it an inflected form, and if so, what is its root form?; does it have known semantic relationships to other words?, etc.

For most collections of material, most token occurrences will be repetitions of tokens seen previously, so caching a working lexicon of tokens seen previously, to record what is known about them, will contribute significantly to efficiency. This is one of the reasons for wanting the tokenizer to recognize the largest possible units that have internal integrity and have some likelihood of subsequent occurrence, so that their first analysis will be shared with repeat occurrences. The second most likely case is for a token to be the first occurrence in the material of a known general English word (assuming we're indexing English). Consequently, if we maintain a relatively large lexicon of known English words, including their inflected forms, then these can be efficiently accessed from an external lexicon at little cost, and if they are then added to the internal cache, this need only be done once for each such token.

In addition to general English words, indexed material may also contain words from specialized or technical vocabularies. Hence, the Pilot system is designed to look up words in a sequence of external lexicons until it finds an entry or runs out of places to look. Thus, the system will typically have a large lexicon covering general English (or whatever language or languages are being used), together with one or more technical lexicons for specialized areas of interest, and perhaps divisional and/or personal supplements for vocabulary unique to an organization or an individual user. These lexicons will contain information about how a word can be used in phrases, and will also contain information about different senses of words and how these word senses are related to other concepts such as more general concepts and semantically-related concepts. This information is used for phrase identification and in concept assimilation.

When all lexical lookups fail, the system applies a powerful battery of morphological analysis techniques to the unknown token, after which the result is entered into the working lexicon of cached tokens so that the analysis does not need to be performed again. This analysis, which we will describe in the next subsection, always returns a conclusion and determines how the system will treat the token. Empirically, with a high likelihood on a wide range of material that we have indexed, when a word is not found in the lexicons, the interpretation assigned by the morphological analyzer is correct. This is partly due to the fact that considerable effort has gone into developing and tuning the morphological rules, and partly due to the fact that equal effort has gone into identifying the cases where otherwise valid rules draw incorrect conclusions. Such cases are then entered into the lexicon.

In fact, the lexicon used by the Pilot system, with the exception of a few thousand words of common English vocabulary and irregularly inflected forms, is devoted mostly to words that valid morphological rules would otherwise analyze incorrectly and words that must be known in order to analyze other words correctly. Words that can be correctly analyzed or guessed by the morphological analysis rules have been systematically omitted. This was done as an experiment in order to see how small a lexicon would suffice when coupled with an aggressive morphological capability. This allows a time/space trade-off for applications targeted for smaller systems. The resulting core lexicon, which can be thought of as an exception dictionary for the morphological analysis component, now has something over 30,000 words (mostly roots and irregularly inflected forms), driven by the process of attempting to give the combined rule system and lexicon the widest possible coverage. For example, a systematic effort has been made to discover and include all of the general English verbs that would not be recognizable as verbs by morphological analysis.

Entries in the lexicon contain information about the possible syntactic categories that a word may have (sometimes with preferences specified), together with various syntactic features such as whether a verb takes indirect objects and what prepositional modifiers it occurs with, and also information about semantic relationships that the word may have to other concepts. It may also contain information about how a word is inflected, how it is capitalized, and variations in its spelling. Often a word may have different senses, and the different senses may have different syntactic categories, different syntactic features, and different semantic relationships. Occasionally, different senses of a word will inflect in different ways. Thus, in the most general case, syntactic and semantic features in the lexicon are associated with word senses, and the lexical entry contains information about the senses that a word is known to have.

Often a word will be highly ambiguous and this information must be taken into account by a phrase analyzer and also by the retrieval algorithm. For example the Pilot system recognizes syntactic category ambiguities such as the following senses of "may":

May as month, May as person's name, may as modal auxiliary,
and word sense ambiguities such as the following senses of "standard":

[adj:standard/normal], [n:standard/flag], [n:standard/criterion]

Note that word senses can be given intuitive names indicating the syntactic category and often a distinguishing subsuming concept. Notice also, that whereas one's first reaction might be to assume that initial capitalization can be used to distinguish "May" from "may," there are many contexts in which capitalization information is itself ambiguous and doesn't contribute to this distinction. This is true, for example, in the first words of sentences, for any words in text that is all uppercase, for capitalized words in titles, and even in some styles of e-mail traffic in which normally uppercase words are left in lower case (as in my e-mail address, usually typed: william.woods@east.sun.com).

I will have more to say about the topic of semantic ambiguity in a separate section later.

3.1.4 Morphological analysis

No matter how complete a lexicon of known words is available, any significant body of indexed material will almost certainly include words that do not occur in the lexicon. This is especially true in a scientific or technical area. Even an unabridged dictionary of English fails to contain many technical terms and many possible words that can be generated from known words by the addition of prefixes and suffixes (despite extensive lists of known cases that are explicitly listed in such dictionaries). Consequently, the Pilot system includes an aggressive morphological analysis component that deals with both prefixes and suffixes as well as regular (and some irregular) inflections. It also deals with lexical compounding (as in "shoelace," "replybuffer," and "bitmap"), and has procedures for guessing the syntactic categories of unknown words based on recognizable endings or lack thereof. In addition, it has a number of special case analyzers for specialized lexical items and formats we have encountered in texts, such as:

date strings—2/18/94, 2-18-94
phone numbers—442-0435
ordinals—21st, 22nd, 23rd, 24th, ...
spelled-out numbers—twenty-three
roman numerals—System V, page iii,
e-mail addresses—william.woods@east.sun.com
underscored words—_U_S_E_R for underlined USER

The task of the morphological analyzer is twofold—(1) to determine a syntactic category and syntactic features for use by a parser to identify and analyze the structure of conceptual phrases, and (2) to identify the root and sometimes other semantic relationships that determine subsumption axioms and other relationships to be added to the taxonomy to support conceptual retrieval. It turns out that taxonomic subsumption, morphological analysis, and word sense ambiguities occasionally intertwine in interesting ways. For example, the suffix *-ful* means one thing when applied to a noun subsumed by the concept [container] (e.g., *cupful*) and a completely different thing when applied to a noun subsumed by the concept [mental attitude] (e.g., *hopeful*). In the first case, it results in a noun that is subsumed by the concept [amount] and in the second case, it results in an adjective expressing a property. The corresponding rules in the Pilot morphological analyzer contain conditions that check the taxonomy for subsumption relations in order to analyze such words correctly.

As of this writing, the morphological analyzer has approximately 600 rules, divided into three categories: (1) inflectional rules that apply to known words that have inflection code properties indicating what paradigm is used to form their inflections; (2) inflectional and derivational rules that apply to a known word based on its spelling; and (3) inflectional, derivational, and category-guessing rules that apply to words with unknown roots or words with no identifiable root. A notable feature of this system of morphological analysis is its ability to analyze words whose roots are not known as well as those with known roots.

In the next few subsections, I will discuss some of the issues in morphological analysis and summarize the capabilities of the current system. If this is too much detail for now, the reader may want to skip ahead to the subsection on phrase extraction and come back here later.

3.1.4.1 Inflection

The system's capabilities for regular (and some irregular) inflections include:

verb inflections:
third person singular (*plays*)
past tense and past participle (*played*)
present participle (*playing*)
archaic second and third person forms (*playest, playeth*)
doubling of final letter (*gas, gasses, gassed, gassing*)
optional doubling of final letter (*focused, focussed*)
final e's (*greeted vs. fated vs. fatted*)
-y to -ies, etc. (*applies, applied, applying, appliest, applieth*)
k insertion (*frollic, frolicked, frolicking*)

noun plurals:

- s, -es, -a, -ae, -i, -en
- y to -ies, doubled letters, final e's, -on to -ata (automata)
- is to -es (theses), -f to -ves (calves), -x to -ces (matrices)
- man to -men (snowmen), alternative -s/es (banjos, banjoes)
- inserted apostrophes (e's)

adjective and adverb inflections for comparative and superlative
(higher, highest, bluer, bluest, fatter, fattest, happier, happiest)

Irregular inflections that nevertheless follow a rule are governed by inflection codes associated with root forms in the lexicon that name the inflection paradigm that the word undergoes.

3.1.4.2 Prefixes and suffixes

In addition to inflectional rules, the morphology component includes derivational rules that analyze words containing recognizable prefixes and suffixes to determine the underlying root from which the analyzed word is derived. Sometimes the rules also derive semantic relationships between the analyzed word and other words (e.g., the root). There are also rules to guess the syntactic category of unknown words based on ending patterns. The Pilot indexer contains rules for analyzing the following affixes and endings:

suffixes and other endings:

- a ability able ae al an ance ant ary
- ate ation atism ative atory borne bred cide color dom
- eater ed ee en ence ency er esque eyed
- fish fly fold ful full graphy ial ian ible ic ical ication id idae
- ify ion ish ism ist ity ive ize less let like ling ly
- man men ment meter most ness ology or ory osis osome otic ous
- ry s safe ship side size some tion tive tomy tor tropic tropy ule ure
- ward wards ware wing wise woman women worm y zoic

prefixes:

- a ab ad aero all allo an ana ante anti auto
- be back bi bio co coco contra counter
- de demi di dia dis disco down em en epi ex extra
- hemi hyper hypo im in inner inter intra iso
- litho maxi meso mini mis mono multi
- neo neuro no non octo off ortho out outer over oxy
- paleo pan para per petro photo poly post pre pro pseudo pyro
- quasi quadri re semi sub super supra
- thermo trans tri un under uni up.

These derivational rules allow for the recognition of forms such as "disruption," "happiness," "renewal," "reunification," and even "antidisestablishmentarianism," without requiring these words to be known in the lexicon, and the rules will relate these forms respectively to the inferred underlying root words "disrupt," "happy," "renew," "unify," and "establish."

3.1.4.3 Prefix-suffix interaction

An interesting problem that arises for a system that analyzes both prefixes and suffixes (most morphological analysis systems used in document retrieval applications do not handle prefixes) is the interaction between the two. Sometimes there is a genuine ambiguity depending on the order in which the prefix and suffix are applied (e.g., untiabable knots, undone shoelaces versus undone packages). The word "untiabable" has one sense (un+tiabable) that means not tiabable and another sense (untie+able) that means able to be untied. (This case is interesting because the senses are almost opposites of each other.) Similarly "undone" has a sense (un+done) that means simply not done, while another sense is the past participle of the verb "undo" and means that the effect of something previously "done" has been reversed. In this case, the interaction is between a prefix and an inflectional transformation, rather than a prefix and a suffix.

Although some words with both prefixes and suffixes are ambiguous in the above way, most such words are correctly analyzed in one way and not in the other. For example, the word "disentanglement" is unambiguously the nominalization of the verb "disentangle" (i.e., disentangle+ment). It is incorrect to interpret it as the removal of an entanglement from something (i.e., dis+entanglement) analogous to "dismember." To correctly model this behavior, the morphological analysis of a new word proceeds in several phases—a first phase for inflectional and suffix analysis of known words, then a phase for prefix analysis if no rules in the first phase succeed, then a phase that looks for lexical compounding (discussed in the next subsection), and finally a second phase of suffix and word-ending analysis if no previous rules succeed. This default ordering of analysis does the right thing with respect to prefix/suffix ordering in the majority of cases, but not always.

To handle cases where the default ordering of prefix versus suffix analysis would not otherwise be correct, some rules contain conditions to block their application if the purported root has specified affixes at the opposite end. For example, the prefix rule for dis- has a condition that will not allow it to apply to a word that has a suffix or that is a past or present participle (so e.g., "disarming" is not dis+arming). These ordering conditions determine the correct behavior in the majority of the cases for which the default suffix-then-prefix-then-suffix-again ordering would not. For the remaining cases, where either the rules would find the wrong ordering or the word is genuinely ambiguous, the offending word is entered into the lexicon as an exception (sometimes, you simply have to know what the correct ordering is). Prefix/suffix interaction is not the only case where you simply have to know how a word is (or is not) analyzed; there are many others. For example, "clarity" comes from "clear" not "clare," "department" is not "depart+ment" (like departure), and "delegate" is not a kind of amputation ((de+leg)+ate). The core lexicon of the Pilot system is heavily oriented towards having the knowledge needed to avoid such mistakes.

3.1.4.4 Lexical compounding

A common pattern of English word formation is to concatenate two or more known words together to form a new word (e.g., shoelace). In English, this kind of compounding is restricted to the coining of new words, unlike in German, where it is a regular part of grammatical sentence formation. Nevertheless, it is a very productive pattern of English word formation, especially in naming plants, animals, and tools, and in modern computer terminology (replybuffer, bitmap, workstation). Consequently, the morphological analyzer contains a component that examines words for this kind of substructure and attempts to

determine a meaning and syntactic category for the compound as a function of the syntactic categories of its elements.

It turns out that there are a number of different patterns for such compound formation, and the current analyzer handles a range of them. Examples of the most common patterns include:

blackberry (noun), a kind of berry
lowflying (adj), a kind of flying modified by low
overflying (noun, adj, and verb), present participle of overfly

Lexical compounding also interacts in interesting ways with apparent participle inflection of nonexistent compound verbs as in:

hardhearted (adj), means with a hard heart (no verb hardheart)
timeworn (adj), means worn by time (no verb timewear)
brickheaded (adj), means with a brick head (no verb brickhead)

Lexical compounding is a kind of analysis that often draws the wrong conclusions, requiring exceptions to be added to the lexicon. However, it is sufficiently productive to make it an important analysis to attempt. For example, on one indexed collection, the query "text search" retrieved a hit on "Dataquest," which was not known to the lexicon, but was analyzed as a compound of data+quest and hence a kind of quest which was known to be a kind of search.

3.1.5 Phrase extraction

In some applications (like the Yellow Pages example), the task of identifying the phrases to be indexed is already done by the application. For full-text indexing, however, a process of phrase extraction must first identify the phrases to be indexed from the running text of the source material. There are a variety of ways to do this, the most expensive of which would be to attempt to parse all of the input material using a natural language parser and grammar. This would be an expensive undertaking, and would not be likely to yield very accurate results with the current state of computational linguistics technology.

The Pilot indexer takes a much more modest approach to phrase extraction by accumulating sequences of words that occur between occurrences of a specified list of "break words" and then analyzing the resulting fragments (which are generally quite short). These fragments are often immediately interpretable as simple noun phrases or verb phrases. Occasionally they contain an extra word or two at the end that must be dropped in order to obtain an interpretable phrase. The system effectively finds the longest initial sequence of each such fragment that can be interpreted as a coherent phrase, and then takes that sequence as a phrase to be indexed.

Some other approaches to phrase extraction have been or are being explored. For example, the Recall system took an even more modest approach by restricting its attention to one-word phrases and simply indexing every content word separately, so as to produce indexes comparable to those of traditional keyword indexing systems. The current phrase extractor for the Nova system is based on a syntactic-part-of-speech tagger [Brill, 1994] that attempts to assign a syntactic category to each word in the text. Phrases to be indexed are then ex-

tracted from the output of the tagger using regular grammars over tags. Other possible phrase extractors for the Nova system are being considered.

A limitation of the approach used in the Pilot system is that the fragments are interpreted in isolation, without any understanding of the sentence context in which they occur. Consequently, an extracted phrase occasionally includes the main verb of a sentence as if it were the head noun of a noun phrase. This happens when the last word of the phrase is one that can be used ambiguously as either a noun or a verb. For example, the phrase "third optional subset lacks" was extracted from an issue of the *Journal of Human Factors in Manufacturing* as if this was a noun phrase whose head word is "lacks" when in fact the word "lacks" was actually the main verb of the sentence from which this phrase was extracted. In order to be safe in such cases, the Pilot indexer will also drop the last word of a phrase and index the shortened result whenever the last word is ambiguous between noun and verb interpretations and the preceding word is also a noun. Thus, in this example case, the correct phrase "third optional subset" is also generated and indexed. Sometimes in these cases the shorter phrase is correct, and sometimes the longer one is. Fortunately, this kind of phrase extraction ambiguity does not happen too frequently, so the major cost of this limitation, the occasional indexing of a spurious phrase in addition to the correct one, is not onerous. For applications in which it matters, the spurious phrases can easily be identified and manually deleted as a post-indexing editing operation.

3.1.5.1 Interaction with markup

One problem for any process that attempts to extract phrases from full text is knowing when to force a phrase boundary because of orthographic features of the source material, such as sentence and paragraph boundaries, the ends of section headings, or the boundaries of table entries. This is where a markup analyzer or implicit markup analysis component becomes important, so as to know when phrase accumulation should continue across an orthographic feature, and when it should be terminated by such a feature. Four particular implicit markup capabilities are included in the Pilot indexer (in addition to its ability to interpret several explicit markup conventions) and are used when the input does not appear to contain any explicit markup:

(1) In text that is not explicitly marked up, short lines that are not last lines of full-sized paragraphs are interpreted as significant, so that phrases will not "flow" past the end of such a line and onto the next line. This simple strategy gives the correct behavior most of the time for a wide variety of situations such as header blocks of e-mail messages. An example is the following text from the annotations at the end of an item in the SunSolve Online™ Systems and Resolutions database:

```
PRODUCT AREA: SunLink
PRODUCT: SNA-P2P
SUNOS RELEASE: any
HARDWARE: any
```

The Pilot phrase extractor correctly extracted the phrases "PRODUCT AREA" and "SUNOS RELEASE" from this passage, while an alternative phrase extractor, that did not recognize the short line convention, extracted two incorrect phrases "SUNLINK PRODUCT" and "P2P SUNOS RELEASE" by continuing to accumulate words into a phrase beyond the

ends of these short lines. (It also destroyed the integrity of the product name SNA-P2P by failing to tokenize it as a single unit and thereby generated a phrase containing only part of this name—another example where tokenizing matters.)

(2) Indentation, which might otherwise signal the beginning of a paragraph, is not considered significant when it occurs immediately after a full line of text that is not indented or is indented less. This simple strategy allows correct interpretation of paragraphs and blocks of text with reverse indentation (i.e., the first line of the block is indented less than the rest of the block). Such reverse indentation is a common convention in some texts. Phrases to be indexed are allowed to flow across the ends of lines followed by such indentation.

(3) Tabs and runs of space characters in the middle of a line that result in vertical alignments of columns between two successive lines are interpreted as field boundaries in a table, so that phrases are not allowed to flow from one field entry to another.

(4) Two or more successive carriage returns or a carriage return followed by white space (other than the reverse indentation convention described above) count as a paragraph boundary or a boundary of a section heading or the beginning of an embedded example, and phrases are not permitted to flow across such boundaries.

Obeying these implicit markup conventions can make a significant difference in the correctness of the phrases extracted from ordinary text files that have not been marked up to indicate their structure explicitly.

3.2 Concept assimilation

Once phrases to be indexed have been identified, a number of activities are involved in assimilating them into a conceptual taxonomy. First, the conceptual structure of the phrase must be determined and represented in order to provide the MSS and MGS algorithms with the information necessary to judge whether one concept subsumes another. This entails parsing the syntactic structure of the phrase and interpreting the result into the representational conventions of the conceptual taxonomy.

Since phrases can be syntactically ambiguous, parsing them sometimes results in several possible interpretations, which would be classified differently. For example, the phrase "Printing Help," which occurs as a section heading in a page of online documentation, could be interpreted either as a verb phrase whose verb is "printing" and whose object is "help" (e.g., printing out help messages) or as a noun phrase whose head noun is "help" modified by the preceding noun "printing" (e.g., help with printing). It turns out that you need to read this section of the manual to figure out which of these interpretations is correct. Since the Pilot indexer doesn't begin to have enough knowledge to correctly resolve this kind of structural ambiguity, it chooses instead to index both interpretations and cross reference them to each other. Thus, it assures that the correct interpretation is in the index at the cost of sometimes including a spurious interpretation.

Interestingly, if the phrase had been "printing helps," there would have been a third possible interpretation—one in which "printing" is interpreted as the subject of the verb "helps" (e.g., it helps if you print). Since this kind of ambiguity is common, and because it is handy to have a concise notation in which to make these different interpretations explicit (espe-

cially when printing out concept names while browsing the conceptual taxonomy), the Pilot indexer uses a convention of enclosing the subject or object of a verb in parentheses when the head of a phrase is interpreted as a verb. This convention can be used to distinguish the verb phrase interpretations from each other and from the default interpretation as an ordinary noun phrase whose rightmost noun (preceding the first occurrence of a preposition, if any) is the head noun. Thus, the three cases are printed as:

printing (helps)—i.e., printing out some things that help
(printing) helps—i.e., it helps if you print something
printing helps—i.e., the plural of a kind of help that is somehow
associated with printing.

In addition to assigning one or more conceptual structures to a phrase and adding them into the conceptual taxonomy at the correct place, the concept assimilator also has the job of determining what other related concepts should be assimilated into the taxonomy, and what relationships should be established between these concepts. For example, in the Pilot indexer, when a noun phrase with more than one modifier is assimilated, then corresponding phrases with one modifier each are generated and assimilated as well. That is, a phrase such as "automobile upholstery cleaning" will also generate the phrases "automobile cleaning" and "upholstery cleaning." These shorter phrases represent more general concepts that will subsume the initial concept and may also subsume other related concepts that will thus be conceptually clustered under these more general categories. An example of this was illustrated in the taxonomy fragment in Figure 1, where the derived phrase "automobile cleaning" subsumed both "automobile upholstery cleaning" and "automobile washing," thereby clustering these two concepts together under a common abstraction that they share.

When a concept phrase has only a single modifier, the Pilot indexer also indexes the head word of the phrase as a single-word concept. Moreover, whenever it adds a single-word concept, it checks the lexical entry for that word and adds to the taxonomy any concepts that the lexical entry lists as subsumers (if they are not already in the taxonomy). It may repeat this operation of adding subsumers until it encounters concepts that are already in the taxonomy or runs out of subsumers. The Pilot indexer also links derived and inflected forms to their root forms as if they were more specific concepts than the root, and it recognizes certain specific kinds of concept (such as numbers, e-mail addresses, and people's names) and links them as individual instances of their corresponding categories. For example, numbers are linked as instances of the concept [number] and alphanumeric strings are linked as instances of a concept [alphanum]. Similarly, a recognized e-mail address such as william.woods@east.sun.com is linked as an instance of [email] and also as a kind of [william.woods].

Finally, when assimilating a concept with modifiers, the Pilot indexer also assimilates each of the modifiers individually and links the concept to each of its modifiers by the relation [modified by]. This relationship is useful to a person browsing the conceptual taxonomy and can also be used by certain retrieval algorithms.

3.2.1 Conceptual structure

In analyzing the conceptual structure of a phrase, the first task is to identify the various roles that different constituents play within the phrase. For example, the determiner in a noun phrase plays a different role from the head noun. The latter is the key conceptual element, while the former adds information regarding uniqueness, focus, quantification, etc. Modifiers play a role that restricts or modifies the meaning of the head noun. In nominalized verb phrases, the object plays a different role from the head verb. The latter is the key conceptual element, similar to the role played by a head noun in a noun phrase. The object of the verb is a modifier of the verb phrase. Determining these syntactic relationships is the job of a parser and grammar. There is a well established body of technique in the field of computational linguistics for parsing natural language sentences and phrases to determine their syntactic structures (see, e.g., [Allen, 1987]).

We can represent the syntactic structure of a phrase by means of a "feature:value" list, a list of feature:value pairs whose "features" include syntactic roles played by various constituents of the phrase (in which case, the "values" are descriptions of the constituents filling those roles). Other feature:value pairs may specify information such as the tense, mood, or aspect of a sentence. The constituents of a phrase may themselves be described by feature:value lists, and so on, resulting in a hierarchical structure of feature:value lists called a *feature structure*. This syntactic feature structure expresses the overall syntactic structure of a sentence or phrase—what its constituents are, what their syntactic features are, and how they are related to each other. For example, the syntactic structure of the phrase "steam cleaning" could be expressed by the feature:value list:

```
head noun: "cleaning"  
modifier: "steam"
```

One or more feature structures of this kind can be automatically derived from a phrase by any of a number of natural language parsers and grammars common in the field of computational linguistics (again, see [Allen, 1987]). In the Pilot indexer, this is done using a parser for ATN grammars [Woods, 1970, 1987]. Note that the words "cleaning" and "steam" are in quotes in the above illustration, indicating that the fillers of these syntactic roles are English words, not the concepts designated by these words.

Another kind of description we can assign to a phrase—the one we are primarily interested in here—is a description that characterizes the "conceptual structure" of the phrase, making explicit the structural information necessary for the MSS and MGS algorithms to test whether one concept is more general than another. Like syntactic structure, conceptual structure can be expressed as hierarchical lists of relation:value pairs (abbreviated r:v pairs). These are similar to syntactic feature structures, except that in conceptual structures there is more structure to some of the r:v pairs. The relation:value pairs in a conceptual structure are of two kinds, expressing respectively, subsumption relationships and modifying constraints. We refer to the former as "governors" and the latter as "modifiers." For example, a conceptual description for the above "steam cleaning" phrase might be something like:

```
kind-of: cleaning  
using: steam
```

In this description, the kind-of relation is used to indicate the governing subsumption relationship that the concept in question is a kind of cleaning. The modifier "using:steam" expresses the constraint that the kind of cleaning being described uses steam. Note that the relations here are semantic, rather than syntactic, and the words "cleaning" and "steam" are no longer in quotes, indicating that what is referred to in these relationships are the *concepts* of cleaning and steam, respectively, not simply the words. This description, unlike the former syntactic feature structure, is not a description of a phrase, but rather a description of a kind of cleaning. These lists of governing and modifying relation:value pairs, describing the structure of a phrasal concept, are called *conceptual descriptions*.

Conceptual descriptions can be of two kinds—*atomic* and *composite*. An atomic description consists simply of the name of an object or category of objects such as cleaning, steam, or Chicago, without any formal structure to such a description. A composite description is a description composed of other descriptions and represented by list of governing and modifying relation:value pairs. We notate such descriptions by enclosing them in square brackets, with the relation:value pairs separated by commas, as in:

[kind-of: cleaning, using: steam]

Indentation can be used to make such expressions more readable, as in:

[kind-of: cleaning,
using: steam]

but formally the indentation adds no additional information to the structure implied by the nesting of the brackets. Producing a conceptual description to represent the conceptual structure of a phrase requires an interpretation of the syntactic structure of the phrase, using the syntactic feature structure and sometimes other knowledge as input.

There is one more technical detail required in order to complete the notation for conceptual description and to fully represent the meaning of our example concept "steam cleaning"—we must specify the intended quantificational import of the modifying relation:value pair "using: steam."

3.2.1.1 Quantificational tags

In constructing conceptual descriptions, it turns out to be important to distinguish various ways to interpret the value of a relation:value pair. Two such ways are: (1) as a name or description for a unique individual filler of the role expressed by the relation, and (2) as a generic description that describes or restricts possible fillers of that role. For example, it is important to distinguish the relationship intended for the name Chicago in "a person from Chicago" (case 1) from that of "big city" in "a person from a big city" (case 2). We can do this by using *quantificational tags* (as in [Woods, 1991]) that can be associated with relation:value pairs in order to specify the quantificational import of a modifier—i.e., the way the value of the modifier stands in the specified relation to the concept being described.

Woods [1991] identifies a number of quantificational tags, corresponding to different ways that a concept referred to as the value of a modifier can stand in relationship to the concept being described by that modifier. Only two of these are used in the Pilot indexer—the tag ME, which expresses the constraint that there exists something of the indicated kind that

fills the role indicated by the relation, and the tag MI, which indicates that the specified value designates a specific individual filler of the indicated role. Mnemonically, the "M" in these tag names stands for "modifier," while the "E" stands for "exists," and the "I" stands for "individual." For most of the examples in this paper, we will use a more concise and intuitive abbreviation of this notation—sometimes using "some" for the tag ME, usually omitting the tag MI when the filler is specified by a named concept specifying the actual filler of the role, and often omitting the tag ME ("some") when the value is specified by a composite description that has a kind-of role within it. Using these conventions, the above two examples could be represented as:

person from Chicago

[kind-of: person, from: Chicago]

person from a big city

[kind-of: person, from: [tag: some,
kind-of: city,
size: [tag: some,
kind-of: big]]]

Note that quantificational tags such as "tag: some" occurring in the specifications of a value for a modifier are functioning as qualifiers on the modifying relationship, not as part of the value description itself.

Using these conventions, we can now fully represent the earlier "steam cleaning" example as:

[kind-of: cleaning, using: [tag: some, kind-of: steam]]

The tag "some" is used because the quantificational import of the relationship "using: steam" is the one expressed by ME ("some") rather than MI. This is because each instance (i.e., occurrence) of the concept "steam cleaning" will presumably use different "instances" of the generic concept steam—not a single individual entity named "steam." (Note: the word "some" has been chosen to read harmoniously both with "mass" nouns, denoting kinds of stuff, like "steam," as well as with ordinary "count" nouns, denoting kinds of things, like "city.") For brevity, we can optionally omit the "tag: some," since this is a composite description with a "kind-of" role. We can thus represent the above example as simply:

[kind-of: cleaning, using: [kind-of: steam]].

3.2.1.2 Conceptual relations

In the modifiers of the above examples, we have specified the relations: *from*, *using*, and *size*, thereby informally introducing the notion of a modifying relationship without any explanation of what it means to be a relation in a modifier. The governing role:value pairs, using the kind-of relation, indicate that a description is a specialization of the specified value concept—i.e., is a subkind of the value concept. Thus, the above description [kind-of: cleaning, using: [kind-of: steam]] is a subkind of cleaning, in the sense that all instances of

this description will necessarily also be instances of the concept [cleaning]. Modifiers in a conceptual description (i.e., the relation:value pairs not specifying a governing subsumption relationship) represent further constraints on the concept being described, usually corresponding to syntactic modifiers in the original phrase.

In forming conceptual descriptions from syntactic structures of English phrases, prepositional phrase modifiers are mapped into relation:value pairs in which the preposition (e.g., "from") is taken to denote an abstract conceptual relation between the thing described by the whole phrase and the value concept corresponding to the object of the prepositional phrase (as in [kind-of: person, from: Chicago]). In a similar way, adjective modifiers are interpreted as specifying a constraint on the value of a conceptual relation that can depend on the adjective. For example, "big" specifies a constraint on the relation, size, and "red" specifies a constraint on the relation, color, as in:

big red apple

[kind-of: apple, size: [kind-of: big], color: [kind-of: red]]

Such modifiers can also be constructed using an open-ended range of conceptual relations corresponding to English transitive verbs, as in:

[kind-of: cleaning, using: [kind-of: steam]]

When interpreting verb phrases, the main verb is taken as the value of a governing kind-of relation, and subjects and objects of the verb are interpreted as values for conceptual relations called "subj" and "obj," respectively, as in:

hunting rabbits

[kind-of: hunting, obj: [kind-of: rabbit]]

Although we will not give details in this paper, it is relatively straightforward to map the syntactic structures assigned to phrases by a natural language parser into the kinds of conceptual descriptions presented here. The resulting conceptual descriptions can then be used to place a concept in the taxonomy by comparing it to other concepts with respect to the relationship of conceptual subsumption.

3.2.2 Conceptual subsumption

As mentioned earlier, the key to conceptual indexing is to organize descriptions in the conceptual taxonomy on the basis of generality—i.e., by subsumption. Informally, a description subsumes another description if the former is more general than the latter. Formally, a description X will be said to subsume another description Y if (1) X has been asserted to be more general than Y in an input "subsumption axiom," or (2) X can be deduced to subsume Y by a chain of transitive subsumption relationships, or (3) X can be deduced to subsume Y by a rule of "structural subsumption," which examines the structures of the two descriptions, and uses subsumption relationships among their parts to judge X to subsume Y if every part (i.e., every relation:value pair) of X can be matched to a corresponding part in Y that is at least as specific. For example, "automobile cleaning" is judged to subsume "car washing" because automobile subsumes car and cleaning subsumes washing.

The criterion of structural subsumption (3) depends on subsumption relationships among the constituents of a concept that may be deduced by any of the three methods. The subsumption relationships derived by transitive closure (2) depend on chains of subsumption relationships of type (1) or (3). Subsumption testing generally bottoms out on a collection of basic facts (axioms) about category inclusion and membership (1) that come from entries made in a lexicon or a semantic knowledge base by a human lexicographer or ontology designer or derived by rule by morphological analysis.

In the taxonomy of the Pilot indexer, there are several ways that a concept can be asserted to subsume another (i.e., be more general than the other). If a noun X is a kind of Y, then Y subsumes X. Similarly, if X is an instance of Y, then Y subsumes X. If X is a more specific verb, or a more specific time, or more specific location, or a more specific name than Y, then Y subsumes X. And if X is an inflected or derived form of an underlying root word Y, or if X is a sense of a word Y, then Y subsumes X. Thus "subsumption" in the Pilot indexer generalizes all of these relations. In particular, subsumption generalizes both the kind-of and instance-of relations. Any of these relations can be asserted into the taxonomy by axioms extracted or derived from a lexicon or as side effects of morphological analysis (or imported from any other source or manually entered directly into the taxonomy). These axioms become the basis for the subsumption conclusions of case (1) above.

Subsumption relationships have characteristic patterns of expression in English, for example:

an X is a Y	e.g., a man is a person
(an) X is a kind of Y	e.g., (a) lion is a kind of animal
X is a Y	e.g., England is a country
X-ing is a kind of Y-ing	e.g., washing is a kind of cleaning

The first two of these patterns correspond to the "kind-of" relation—they say that a category X is a subkind of a category Y (i.e., X is kind-of Y). The third corresponds to the "instance of" relation. It says that an entity X is an instance of a category Y (i.e., X is instance-of Y). The fourth pattern asserts that an action X is a subkind of an action Y (e.g., WASH is kind-of CLEAN). Some people object to calling this relationship between verbal actions "kind of," preferring to reserve that terminology for a relationship that holds exclusively between nouns. Such semantic fastidiousness doesn't decrease the value of treating this relationship as a kind of subsumption relationship for purposes of organizing a taxonomy. If it makes you feel better, you can think of this as a special relation, different from "kind-of," that is another specialization of the relational abstraction "subsumed-by" and that holds between verbs.

Note that the relation kind-of is a *transitive* relation, analogous to the subset relation between sets (or equivalently, the inclusion relation in Venn diagrams). That is, if X is a kind of Y and Y is a kind of Z, then X is a kind of Z. Similarly, the root-of and sense-of relations are transitive. However, instance-of is an intransitive relation, analogous to set membership of a item in a set. That is, if X is an instance of Y and Y is an instance of Z, then it does not follow that X is an instance of Z. For example, we would say that Captain Ahab is a ship captain, and that ship captain is a profession, but not conclude that Captain Ahab is a profession.

3.2.2.1 Relational abstraction

In a previous example, we interpreted "big" as constraining the value of a relation: size. But note that size is a generalization of many other attributes such as area, volume, height, weight, girth, etc., any one of which, or some abstract combination of which, might have been the intended sense of "big." In the case of a big city, the possible relations are probably population-size or geographical-area. Thus, our interpretation is a generalization of several more specific interpretations that might have been intended. Other adjectives, such as tall and thin, specify more specific subkinds of size. Thus not only concepts, but also relations, can have subkinds and can subsume one another. Correspondingly, composite descriptions can subsume one another by virtue of the subsumption of relations. For example,

[kind-of: person, size: [kind-of: large]]
subsumes
[kind-of person: height: [kind-of: large]]

Handling this kind of relational abstraction turns out to be an important capability that we can exploit to solve a number of problems in natural language interpretation and paraphrase matching. For example, in the steam cleaning example above, we interpreted the phrase "steam cleaning" to be:

[kind-of: cleaning, using: [kind-of: steam]].

However, this interpretation requires general world knowledge to determine that the relationship between steam and cleaning in this case is "using." Many other such relationships are possible for nouns modifying other nouns, including made-of (e.g., in "aluminum pot"), used-for (e.g., in "horse shoe"), floats-on (e.g., in "water craft"), and so on, almost without end. In some cases, it is difficult to even give a short name to the relationship involved (e.g., in "door bell").

In ordinary language understanding, we interpret such phrases by finding a suitable relationship that we know makes sense and interpreting the phrase accordingly. Sometimes there are several possible relationships that would make sense, in which case the phrase is either interpreted as ambiguous or one of the senses is known to be conventionally used and is therefore preferred over the others. For example, the phrase "alligator shoe" might be interpreted as a shoe for an alligator (like a horse shoe) if it were not known by convention to be a shoe made of alligator skin.

To deal with this in a system that will not usually have the necessary knowledge, we will make use of a very abstract relationship called "mod" that subsumes all other relationships. That is, any relationship (e.g., "using") is a kind of mod. Prenominal noun modifiers (i.e., noun modifiers that occur before the head noun) will then be interpreted as standing in this abstract relation (mod) to the described object. Such descriptions will then subsume any description using a more specific relationship. That is:

[kind-of: cleaning, mod: [kind-of: steam]].
subsumes
[kind-of: cleaning, using: [kind-of: steam]]

because mod subsumes using (by definition of mod).

Similarly, for an adjective with no obvious corresponding relation, or whose corresponding relation is unknown, we can represent the relationship by using the relation mod.

The concept of relational abstraction can also be used to clarify the relationship between the concept of subsumption and the relations kind-of and instance-of. Specifically, kind-of and instance-of can be thought of as subkinds of a common abstract relation: subsumed-by. Thus subsumed-by subsumes kind-of and instance-of.

Relational abstraction can also be used to capture a range of paraphrase relationships among different ways to describe actions. Specifically, by making the relations [subj] and [obj] specializations of the preposition [of], which in turn is a specialization of the abstract relation [mod] (by definition of mod), we can systematically relate phrases such as "lion hunting," "hunting of lions," "hunting (lions)," and "(lions) hunting" in the taxonomy:

```
lion hunting i.e., [kind-of: hunting, mod: [kind-of: lion]]
|-k- hunting of lions i.e., [kind-of: hunting, of: [kind-of: lion]]
   |-k- hunting (lions) i.e., [kind-of: hunting, obj: [kind-of: lion]]
   |-k- (lions) hunting i.e., [kind-of: hunting, subj: [kind-of: lion]]
```

Figure 3. A hierarchy of sense ambiguity relationships.

Legend: In the notation of this display, the symbol |-k- represents a kind-of link from the indented concept to the concept at the head of the vertical bar. Similar links with different internal letters can be used to represent instance-of links (i), links to morphological roots (r), and links from word senses to word concepts or more general word senses (s).

The taxonomy in Figure 3 nicely captures (in a way that can be structurally exploited) the ambiguity of the phrase "hunting of lions" between the two more-specific senses in which the lions are either the subject or the object of the hunt, and it relates all of these to the more abstract concept "lion hunting" (which could also subsume other concepts, such as, for example, hunting the way lions do).

3.2.3 Organizing the taxonomy

We now have a formal way of deciding if one description is more general than another and a way to use this relationship to capture and exploit subtle paraphrase variations among different ways to express concepts. This relationship can now be used to organize conceptual descriptions into a structured taxonomy that can be used for efficient search and retrieval by an MSS algorithm. In order for this algorithm to use a conceptual taxonomy to efficiently discover where in the taxonomy a new (query) concept would belong, it is necessary to first organize the concepts into a structure in which every concept has explicit pointers to its most specific subsumers and from its most general subsumees. This provides explicit

links for a minimal collection of subsumption relationships from which the rest can be found efficiently by transitivity—i.e., by following sequences of MSS links through the graph. (This process of following sequences of links through a graph-based knowledge representation is sometimes referred to as "pointer chasing" or "path-based inference.")

Because chains of MSS relationships moving upward through a taxonomy tend to converge at the top, following such chains is an efficient way to test for any subsumption relationship implied by the taxonomy and is used as the basic algorithm for testing subsumption of constituents when comparing a new description with existing descriptions in the taxonomy. Moreover, the structure of the taxonomy can be used as a way to search for the location where a new description belongs (i.e., below its MSSs and above its MGSs). The situation is somewhat analogous to the use of a binary search algorithm, which requires that the collection to be searched must first be organized into a numerical or alphabetical order that can be tested. In this case, however, the ordering involved (i.e., subsumption) is a partial order, rather than a total order, so the structure to be searched cannot be a single ordered list. Instead of having a single predecessor and a single successor, as each element would have in a total ordering, the concepts in a conceptual taxonomy may have several predecessors (MGSs) and several successors (MSSs).

Organizing the taxonomy is a process that can be thought of as a kind of *conceptual sorting* that produces a partially-ordered structure in which each concept points to its most specific subsumers and is pointed to by its most general subsumees. One way to achieve this organization is to incrementally add concepts into the taxonomy one at a time, each time placing the inserted concept in the correct position linked to its most specific subsumers and most general subsumees (somewhat analogous to an insertion sort), beginning with an initial taxonomy consisting of the asserted subsumption relationships (axioms) among the atomic concepts. Adding a new description to the taxonomy involves three steps:

1. Use the MSS algorithm to find the concept's most specific subsumers in the taxonomy.
2. Use an analogous MGS algorithm to find the concept's most general subsumees in the taxonomy.
3. Add kind-of links to the graph structure from the concept to each of its most specific subsumers and from each of its most general subsumees to the concept, and then eliminate links from the structure that are now redundantly implied by transitivity of the kind-of relation—i.e., those between one of its most general subsumees and one of its most general subsumers.

If the taxonomy starts out properly organized (i.e., with each concept linked appropriately to its most specific subsumers and most general subsumees), then this insertion process will preserve that property as the taxonomy grows. More details of the operation of the MSS and MGS algorithms are discussed in [Woods, 1991]. Both theoretical arguments and empirical experience with our implemented algorithms indicate that at least for the kinds of conceptual description used here, these operations can be done on the average in an amount of time that is sublinear with respect to the size of the taxonomy.

4 Navigation and Retrieval

The taxonomic structure of a conceptual index provides an efficient network of pathways that can allow a person to navigate through conceptual space and can also enable retrieval algorithms to efficiently find relationships between concepts. It can support human browsing and navigation in "conceptual space" by providing a structured map of the concepts used in the indexed material and allowing a user to move conveniently back and forth between concepts in the taxonomy and places in the text material where those concepts occur. It can also support text search algorithms that can use paths in the conceptual index to find relationships between terms in a request and related terms that may occur in relevant material.

One way the conceptual taxonomy can be used to support navigation is to start with one of a collection of topmost nodes in the taxonomy, called the *roots* of the taxonomy. (Imposing a constraint that there be a single topmost root that subsumes all other concepts is a design option, but this is not necessary.) Any concept in the taxonomy can then be found by beginning with some root and proceeding down through a sequence of subsumption links to the desired concept. However, this way of navigating (which is possible in any hierarchical navigator) is not a significant feature of a conceptual index, and the conceptual taxonomy is seldom used in this way.

Unlike any other kind of hierarchical navigator, the browser of a conceptual index can jump immediately into the conceptual "neighborhood" of any specified request, without having to follow a sequence of successive links from a base root, and without requiring that the request specifically name a concept in the taxonomy. This is because the formal taxonomic structure of the conceptual taxonomy provides a *topology* for conceptual space and supports an algorithm to find concepts in the "conceptual neighborhood" of a requested description even when that description does not itself occur in the taxonomy. The MSS algorithm will automatically locate the most specific subsumers of any requested description that does not already occur in the taxonomy. Functioning as the "transporter room" of this "Enterprise," the MSS algorithm can move an information seeker directly to the neighborhood of any specified request.

Once focused on a conceptual neighborhood of interest, an information seeker can move upwards and downwards through a conceptual taxonomy by following MSS and MGS links. Thus, the structure of a conceptual taxonomy provides an orientation in conceptual space by distinguishing "up" (more general) from "down" (more specific) relationships. It also provides a notion of "sideways" (i.e., sibling with respect to MSS) relationships when two concepts have the same MSS "parent." A conceptual taxonomy also provides intuitively meaningful descriptions of each concept in the space that relate formally to these orientations, so that a human browser is less likely to lose track of his or her location and orientation in the taxonomy. Concepts may also have additional (nontaxonomic) links, representing other relationships to other concepts, which can also be followed through the taxonomy. For example, in the Pilot system, a concept has "mod" links to other concepts that modify it and "reverse mod" links to other concepts in which it is a modifier.

While browsing a conceptual index, an information seeker can at any point follow links from a concept in the taxonomy to all of its occurrences within the indexed material and from any concept in the material to the neighborhood of that concept in the taxonomy.

Thus, an information seeker can move conveniently back and forth between concepts in the taxonomy and their occurrences in the text material. This allows the taxonomy to be used as a powerful tool for navigating the text material.

4.1 Dynamic passage retrieval

While the links from a concept in the taxonomy to the places where it occurs in the indexed material can be used to locate passages in the material where a concept occurs literally, it is often the case that a concept is implicitly present in a passage of text without occurring literally. For example, the phrase "brown coat" does not occur literally in the sentence "The coat was reddish brown," but it is implied by what is there. In this case, a relatively straightforward transformation on a parse tree could annotate the sentence with a (virtual) occurrence of this concept and thus establish a connection. Other cases, unfortunately, involve a great deal of reading between the lines and subtle combination of evidence in order to determine that a requested concept is implicit in a passage. The necessary evidence may even be spread over several sentences. Worse, these tougher cases tend to outnumber the easy ones.

In order to deal with circumstances in which a concept is implicit in a passage but not literally present (another manifestation of the paraphrase problem), the experimental Pilot system was equipped with a search capability that attempted to locate passages where a requested piece of information might be implicitly present, and to score and rank such passages on the basis of an estimate of the degree to which the passage looked promising. This search capability was based on a different perspective from traditional information search and retrieval and has turned out to be a major contribution all by itself. We call this approach "dynamic passage retrieval," and we call the specific technique we use to identify, score, and rank passages "relaxation ranking."

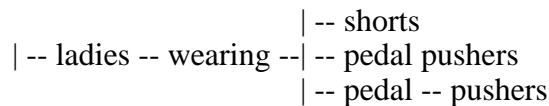
The essential idea of the *relaxation ranking* retrieval algorithm is to use the concept occurrence position information from the conceptual index to find places in the indexed material where all (or as many as possible) of the elements of a request are represented as near to each other as possible (and as nearly in the same order, or as nearly in the same syntactic or conceptual relationship, and as nearly in the same terms, and using as nearly the same inflected forms, as possible). The degrees to which an identified passage differs (i.e., the match criteria must be "relaxed"), in each of these dimensions of relaxation, from a direct literal occurrence of the requested concept are then combined into a weighted penalty score that is used to rank the resulting passages. The passages with the lowest values of this penalty-based *relaxation ranking* score are ranked first (i.e., better), and the resulting passages are presented to the information seeker in rank order, together with their relaxation ranking scores and the justifications for their match. Hypertext links from the items in this results display to the actual locations of the passages in the indexed material and links from elements of the results display to the corresponding concepts in the conceptual taxonomy complete the human interface.

Unlike traditional information retrieval (which normally retrieves whole documents), a dynamic passage retrieval algorithm retrieves and ranks individual passages within documents that are judged likely to contain the information sought. Moreover, unlike most other passage retrieval systems that determine a fixed level of passage "granularity" (say, a paragraph) at the time of indexing and then index such passages as if they were small doc-

uments, a *dynamic* passage retrieval algorithm uses position information of the occurrences of concepts to identify passages at retrieval time. Moreover, the granularity of the resulting passages is variable, depending on how closely together the elements of the request can be found together.

4.1.1 The relaxation ranking retrieval algorithm

Dynamic passage retrieval using the relaxation ranking retrieval algorithm begins with an interpretation of the input request as a sequence of concept descriptions, each of which would be matched successively in that order in an ideal target passage. The simplest such analysis is to interpret the input request as a sequence of one-word concepts, one corresponding to each term in the request. A more complex interpretation would parse the entire request and generate a conceptual description for each plausible interpretation of the request phrase. Failing a complete analysis of the request as a single phrase, all plausible segmentations of the request as sequences of analyzable phrases could be generated. The most general case is to interpret the input request as a directed acyclic graph of alternative sequences of concept phrases to be sought, each phrase of which may have one or more interpretations expressed as conceptual descriptions. We will call such a graph a "request graph." For example, a request such as "ladies wearing shorts or pedal pushers," might be transformed into the request graph:



corresponding to the sequences of concepts:

```
ladies -- wearing -- shorts
ladies -- wearing -- pedal pushers
ladies -- wearing -- pedal -- pushers
```

where in the second sequence, the phrase "pedal pushers" is interpreted as a single concept, which in this case would (if correctly interpreted) be an atomic concept known to the taxonomy as a kind of ladies pants (as opposed to the structured concept: [kind-of: pushers, mod: [kind-of: pedal]]). If the system were as knowledgeable as you and I are and as skilled in resolving intentions behind utterances, then the third sequence here would presumably have been rejected, but let us assume here (plausibly) that our system is not that clever.

The relaxation ranking retrieval algorithm attempts to find compact passages of source material where one of these sequences of concepts is matched to some degree by a corresponding sequence of concepts in the material. The matching in question involves a graded relaxation of the criterion that the sequence of concepts should be matched in the source material in the exact form and wording and in the exact order and with no intervening words or sentence boundaries. The details of the algorithm for doing this efficiently are complicated and will not be presented here. However, the essential idea is to create generators for each of the concepts in the request graph that will enumerate (in order) all of the places in the source material where those concepts occur. For example, the generator for the concept "ladies" would generate all of the occurrences of that concept in the collection in order of their occurrence in the source material. In doing so, it would generate not only occurrences

of the word "ladies," but also occurrences of any concept known in the taxonomy to be more specific than ladies, including the names of any known ladies and any more specific category of lady, such as "begum," which is known in the lexicon to have a sense (as a noun) that is a kind of lady of high rank. In the Pilot version of the algorithm, all inflectional variations of these terms are also generated, but with those in the plural in this case (the form of "ladies" in the request) preferred.

Having found an occurrence of one of the concepts in a request sequence at some point in the source material, the algorithm looks for occurrences of other concepts in the sequence within a maximum distance (specifiable as a parameter to the search) and generates and scores candidate passages for any set of such source concepts that matches the request sequence. The scores assigned to a candidate passage are the accumulation of penalties for various dimensions of relaxation from an ideal exact match of the request sequence. If no candidate passage containing representatives of all the elements of the request sequence are found, then candidate passages for as large a subset of those elements as possible are generated and scored. Any candidate passages that overlap another such passage with a better score (i.e., lower penalty) are suppressed, and all remaining passages are merged into the results list in order of their penalty scores (and in order of their positions in the source material when the scores are tied). If there is a specified maximum number of hits to be returned in a results list, then candidates whose scores push them off the bottom of the list can be dropped, and if the results list is filled with perfect scores (zero penalties) then the entire process can stop. Otherwise, even when the results list is filled, search has to continue looking for better hits that may displace something from the list.

A more complex interpretation of the above example, would include two additional request sequences, consisting of a single concept phrase each, for the concepts "ladies wearing shorts" and "ladies wearing pedal pushers," giving the resulting directed acyclic graph:

```

    | -- ladies wearing shorts
    | -- ladies wearing pedal pushers
--- |                               | -- shorts
    | -- ladies -- wearing --| -- pedal pushers
                               | -- pedal -- pushers

```

In this case, concepts in the text directly subsumed by the concept descriptions:

```

[kind-of: ladies, mod: [kind-of: wearing, obj: [kind-of shorts]]]
[kind-of: wearing, subj: [kind-of: ladies], obj: [kind-of shorts]]]
[kind-of: ladies, mod: [kind-of wearing, obj: [kind-of: (pedal pushers)]]]
[kind-of: wearing, subj: [kind-of: ladies], obj: [kind-of (pedal pushers)]]]

```

would be returned as well as those for the previous interpretation, assuming that each of these phrases has two assigned conceptual descriptions, one with "ladies" as head and one with "wearing" as head. Of course, the likelihood of finding a literal subsumption of these specific concepts may be less than that of finding the separate individual words near each other, but when such subsumptions occur, they will be exact matches, even if the terms involved in the matching phrase are different or in a different order, for example: "modeling

of neon pedal pushers by society matrons."

Even for the case where the interpretation of a request is simply a single sequence of one-word concepts, the above passage would be discovered, but wouldn't be given as good a score. The conceptual taxonomy would still be used to find words and phrases subsumed by the concepts of the request. In the above example, a sense of "modeling" would be subsumed by "wearing," "pedal" and "pushers" would be subsumed by equality, and "society matrons" would be subsumed by "ladies." The penalty score associated with the passage, however, would now reflect the presence of the intervening word "neon" and the fact that the concepts occur in a different order.

A slightly better result would be achieved by the first analysis of the request given above, which recognized the existence of the known lexical phrase "pedal pushers." In this case, the phrase "neon pedal pushers" would be subsumed by the phrase "pedal pushers," and so "neon" would no longer count as an intervening word.

The penalty scores assigned by the relaxation ranking retrieval algorithm are determined by a set of parameters that specify the penalties to be added for an intervening word, for an inflectional difference, for a proper subsumption match (as opposed to a same term match), for terms being out of order, for a term to be missing, etc. One can imagine training this system on a large number of hand-annotated examples in order to determine "optimal" values for these parameters (at the risk of "overtraining" to artifacts of the training set that don't generalize). However, even with an untrained, intuitive, seat-of-the-pants choice for these weighting parameters (i.e., with no training of the system to determine optimal values), and with a modest lexicon and small set of subsumption axioms, the ranking results of this technique on the first few domains on which we tried it were noticeably superior to those I had experienced with other relevance ranking retrieval systems. The first ranked passage was so often correct that the Pilot system was configured to automatically go directly to the location of the first ranked passage and to step through successively ranked passages at the touch of a button. This is in contrast to most retrieval systems, which first present a list of hits and wait for the user to specify which ones to explore. In the Pilot system, such a list of results is only displayed when the user requests it from a menu of options.

The Recall system was designed and implemented to explore the capabilities of the relaxation ranking algorithm for dynamic passage retrieval, and to compare its abilities with those of traditional document retrieval systems. The results of that exploration, which I will take up in the next section, were sufficiently positive that we now use the term "Precision Content Retrieval" to refer to the combination of conceptual indexing with dynamic passage retrieval using the relaxation ranking algorithm. Precision content retrieval is able to deliver a level of access to information that can be thought of as falling between that of *document retrieval* and *fact retrieval*. Fact retrieval systems (also called question answering systems) deliver specific answers in response to specific questions that are fully analyzed both syntactically and semantically and processed against a fully analyzed database. Such systems are usually supported by structured databases rather than unstructured text. Precision content retrieval can deliver some of the functionality of a fact retrieval system by identifying specific passages where the requested information is likely to be found, but without the requirement of having a codified structured database.

5 Experience and Evaluation

The knowledge technology group in Sun Microsystems Laboratories began this project by looking at the needs of users of online documentation systems such as Sun's Answer-Book™ and the UNIX® man pages. The hypothesis was that techniques from taxonomic knowledge representation and natural language processing would help. We began by looking at the problems that real users encounter when using online information systems. I attempted to catch people in the middle of an information need and then document how they expressed their need spontaneously and how the material they eventually found was expressed. We quickly focused on the paraphrase problem, a major stumbling block that often stymies a user who does not have a good intuition about how terminology is used in the material being searched. This is especially a problem for the users that need to use online documentation the most—new users. The Pilot system is dedicated to them.

The first (and continuing) experimental conceptual indexing system, now called Pilot, was implemented in Lisp by lashing together a basic free-text phrase extraction algorithm (described earlier) with some natural language processing technology I had developed previously and with a taxonomic classification algorithm written by a summer intern. This system was (and still is) fairly slow and limited to bodies of material whose indexes can be stored in a memory image, but it is powerful and easy to change, and it has now been applied (and continues to be applied) to a wide variety of material in order to ensure that the technology is robust and relatively domain-insensitive. The illustrative examples of the technology presented in the next section come from Pilot explorations.

The Recall system was implemented specifically to explore the performance of dynamic passage retrieval technology (specifically the relaxation ranking retrieval algorithm) and compare it to the performance of traditional information retrieval systems. Some of the results of that experimentation are presented in the upcoming section on formal measurement. The Solar system, which operates as a network server with a Netscape™/Mosaic interface, was implemented to explore the reactions of real users to this technology. Experiences of various kinds of users with this technology are covered in the section on "the user experience."

5.1 Illustrative examples

The power of a conceptual indexing system to address the paraphrase problem can be illustrated by the following results from an experiment in which we indexed a collection of computer bug reports. The query "color change" retrieved the following phrases extracted from bug descriptions:

"becomes black"
"reset bitmap colors"
"color disruption"

These descriptions were among those automatically extracted from the bug report database and automatically assimilated into the conceptual taxonomy by the conceptual indexing algorithm of the Pilot system. They were found in response to the query by classifying the query and displaying the most general subsumees of the query using the MGS algorithm.

These examples show the results of using subsumption axioms for kind-of (e.g., become is a kind of change, reset is a kind of change) and instance-of (i.e., black is an instance of color). The second example uses a morphological subsumption fact ("color" is the root of "colors") to make its connection to the query. Both of the first two phrases show the benefits of analyzing phrases with a parser and lexical knowledge about syntactic categories of words in order to derive their conceptual structure, since both are verb phrases rather than noun phrases and their head words are not the rightmost. Using the parenthesis conventions for specifying verb phrase structure, these would be represented as:

"becomes (black)"
"reset (bitmap colors)"

The third example illustrates the interaction of morphological and taxonomic knowledge, since it results from morphologically analyzing "disruption" as a form of the verb "disrupt," which is known to be a kind of "damage," which is known to be a kind of "change."

These examples also illustrate how different facts and features make a difference for different queries and different hits. Contemplation of these examples suggests that it is difficult to try to anticipate in detail what facts and features are going to be important for some future query, other than to systematically catalog and codify basic subsumption facts that seem to have general potential utility. These examples would be totally uninteresting if the facts and features required to deal with them were developed after seeing the problems. What is interesting is that a general strategy of incorporating certain kinds of basic facts and features ended up finding examples like these in a totally unanticipated combination of text and query.

5.1.1 Conceptual clustering

One of the attractive features of a conceptual index is that the conceptual "neighborhoods" determined by the collection of subsumees of a concept constitute a meaningful hierarchical clustering of concepts that can be automatically derived from unrestricted text. These neighborhoods (the subsumees of a concept) gather together in one place a group of semantically-related phrases that occur at various places in the source material. Moreover, this methodology gives the resulting clusters a linguistically meaningful characterization of what the elements of the cluster have in common and the level of generality of the cluster—i.e., the concept at the top of the cluster. This is in marked contrast to other clustering techniques, which often produce only the set of elements of the cluster. Some techniques produce an abstract prototypical member, such as the center of mass in some feature space, or a statistically-derived summary, such as a list of the most frequently occurring words. A clustering based on a distance measure characterizes the level of generality by the threshold distance value that determines the "radius" of the cluster, and a center-of-mass prototype element, by itself, gives no characterization of the level of generality or size of the corresponding cluster. In a center-of-mass-based system, the same prototype element could be the center of mass for each of a chain of "concentric" clusters.

An example of conceptual clustering is given in Figure 4, which illustrates a hierarchical semantic cluster of phrases derived by conceptually indexing an issue of the *SunExpress*TM product catalog. The fragment of taxonomy in the figure is a hierarchical cluster of phrases having to do with the concept of adding memory. Phrasings using the words "memory,"

"storage," and "disk," together with "add," "additional," "adding," and "soldered-in," are pulled together in one structured conceptual neighborhood. (In case you're wondering, "soldered-in" was analyzed as having the root "solder," which was known to be a kind of "attach," which was recorded as a kind of "add.")

One interesting observation in Figure 4 is that some verb phrases (e.g., "purchase additional memory") are indexed as if they were noun phrases, clearly an incorrect analysis. (Recall that the Pilot system generates alternative analyses of such phrases and cross references them when it doesn't know enough to choose the correct analysis.) The interesting thing about these cases is that, contrary to the expectation that such mistakes might cause problems, typically these additional cross references are actually useful. If the system resolved these ambiguities, then the value of this conceptual cluster might actually be diminished.

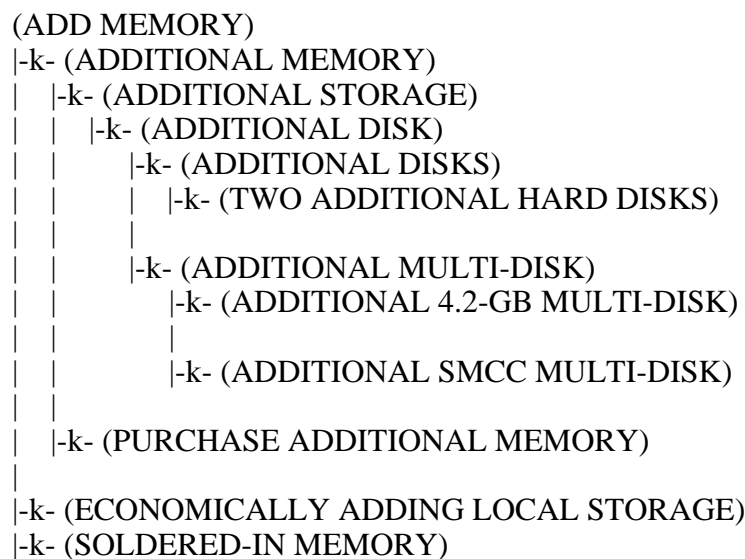


Figure 4. A taxonomy fragment from a product cat

For the record, this issue of the SunExpress catalog contained:

- 10,284 database files
- 8.6 megabytes of data
- 1,097,899 words of text
- 813,573 indexed content word occurrences
- 20,563 unique words
- 51,525 sentences / section headings / isolated phrases
- 101,812 distinct concepts.

5.1.2 The power of subsumption axioms

An interesting bit of data resulted from indexing the SunExpress catalog. When this catalog was first indexed, all that the lexicon knew about the word "disk" was that it was a flat circular object. Consequently, in order to see what effect more knowledge would have, I added a new axiom to the conceptual taxonomy, expressing the fact that a disk is a kind of secondary memory storage. The Pilot indexer includes a capability to accept new axioms at any time and then derive the additional subsumption relationships enabled by each new axiom. When it does this, it prints out a record of the new relationships that it finds. Using this capability, it was possible to assess the impact of this single axiom. Figure 5 illustrates some of the connections that were made between otherwise unrelated phrases in the catalog material as a result of adding the single axiom: DISK is kind-of (SECONDARY MEMORY STORAGE).

(LOCAL DISK) is kind-of (LOCAL STORAGE)
(DISK SPACE) is kind-of (STORAGE SPACE)
(AVAILABLE DISK) is kind-of (AVAILABLE STORAGE)

(MULTI-DISK SYSTEMS) is kind-of (STORAGE SYSTEMS)
(8.4-GB OF HIGH-PERFORMANCE DISK)
 is kind-of (8.4-GB OF STORAGE)
(NEW 2.1-GB DESKTOP DISK PACK)
 is kind-of (2.1-GB DESKTOP STORAGE PACK)

... and many more ...

including some amusing, benign mistakes:

(COST OF PRODUCING CD-ROM DISK)
 is kind-of (STORAGE COST)
(WIDE DISK) is kind-of (LARGE STORAGE)

Figure 5. Relationships generated by a single axiom.

5.1.3 Navigating in conceptual space

As mentioned earlier, one of the useful things about conceptual taxonomies is their support for navigating in conceptual space. The utility of browsing in conceptual space can be illustrated by the following example from a conceptual index of a collection of encyclopedia articles about animals. An initial request for "brown fur" produced the display shown in Figure 6. This display shows the more specific concepts in the taxonomy that are subsumed under the concept (BROWN FUR) followed by a display of the conceptual structure of the phrase (a computerized version of the [kind-of: fur [mod: [tag: ME, kind-of: brown]] notation described previously). This is followed by a display of the more general concepts above (BROWN FUR) in the taxonomy, which is followed by a listing of directed forward

links leaving the node (a mod relationship to the concept brown in this case) and directed reverse links entering the node (none in this case).

```
Show (BROWN FUR):

(BROWN FUR)
|-k- (GRAY BROWN FUR)
|-k- (RICH BROWN FUR)
-k- (WHITE-SPOTTED BROWN FUR)

(NOM
 (KIND-OF FUR)
 (ME MOD BROWN))

More general concepts:
kind-of (BROWN COAT)
  kind-of COAT
    kind-of SOMETHING
kind-of FUR
  kind-of ANIMAL_COAT
    kind-of (ANIMAL COAT)
      kind-of COAT
        kind-of SOMETHING
Forward links:
  MOD BROWN
Reverse links:
```

Figure 6. A taxonomy browser display, showing more general and more specific concepts.

The concepts directly above (BROWN FUR) in the taxonomy come from structural subsumption and from an axiom in the lexical entry for FUR that says that FUR is kind-of ANIMAL_COAT. Here, the common technique of connecting words with an underscore is used to denote a phrase in the lexical entry. The morphological analyzer in the Pilot indexer recognizes the internal structure of the "word" ANIMAL_COAT and so generates a kind-of relationship from it to the concept phrase (ANIMAL COAT). The other relationships to FUR and COAT are generated in the concept assimilator by recording subsumption relationships from two-word phrases to their head words —i.e., (BROWN FUR) is a kind of FUR, (BROWN COAT) is a kind of COAT, and (ANIMAL COAT) is a kind of COAT. The structural subsumption of (BROWN FUR) by (BROWN COAT) is inferred by the fact that BROWN subsumes BROWN, and COAT subsumes FUR by virtue of the chain of kind-of relationships from FUR to ANIMAL_COAT to (ANIMAL COAT) to COAT.

```

(BROWN COAT)
|-k- (BRIGHT REDDISH BROWN COAT)
|-k- (BROWN BLACK COAT)
|-k- (BROWN COATS)
|   |-k- (FAWN COATS)
|   |   |-v- ((FAWN) COATS)
|   |
|   |-k- (REDDISH BROWN UPPER COATS)
|
|-k- (BROWN FUR) ...
|
|-k- (BROWN HAIR)
|   |-k- (BROWN HAIRS)
|   |   |-k- (REDDISH BROWN GUARD HAIRS)
|   |
|   |-k- (BROWN WOOL)
|   |   |-k- (REDDISH BROWN WOOL)
|   |
|   |-k- (BROWNISH HAIR)
|   |-k- (REDDISH BROWN HAIR)
|       |-k- (REDDISH BROWN GUARD HAIRS)
|       |-k- (REDDISH BROWN WOOL)
|
|-k- (BROWN-GRAY COAT)
|-k- (BROWNISH COAT)
|   |-k- (BROWNISH HAIR)
|   |-k- (BROWNISH SUMMER COAT)
|   |-k- (COARSER BROWNISH COAT)
|   |-k- (DARK BROWNISH COAT)
|
|-k- (COAT OF BROWN)
|   |-k- (WOOLLY COAT OF REDDISH BROWN)
|-k- (TAWNY COAT)

```

where the ... indicates the subtree already shown in Figure 6.

Figure 7. A more general request, illustrating the utility of navigating in conceptual space.

When I first posed this request for "brown fur" and saw the display in Figure 6, I noticed the more general concept (BROWN COAT) in the display of more general concepts and recognized that I had an opportunity to find more examples. The Pilot indexer was designed to provide this display of more general concepts for exactly this kind of situation, and I always check the more general concepts to see if there is one that more accurately expresses what I am looking for. In this case, I immediately generalized my request to (BROWN COAT), which more correctly captured the level of generality I was interested

in. This resulted in the substantially more complete display shown in Figure 7. Thus, by displaying the more general concepts in the taxonomy above the level of the stated request, the system unobtrusively suggested a useful generalization of the request.

5.2 Formal measurements

As mentioned previously, when the relaxation ranking dynamic passage retrieval algorithm was incorporated into the Pilot system, the results seemed so good that we designed the Recall system to explore the retrieval capabilities of this algorithm and to compare its capabilities with the capabilities of traditional information retrieval techniques. We indexed a snapshot of the UNIX man pages, and we gathered a collection of queries that were real statements of information need extracted from a log book kept by a new user of the Sun operating system and expressed in the language spontaneously used by that user to state the need. These queries, which are generally short and quite specific, represent an extremely difficult challenge for traditional information retrieval techniques. Once these queries were chosen, an experienced UNIX user determined a set of man pages that would be considered acceptable answers to each query. These judgments became the standard against which the systems being compared would be judged.

For these experiments, the output of the Recall system was made comparable to that of traditional document retrieval systems by scoring each retrieved document using the score of its best contained passage. (If it contained no such passage, it wasn't retrieved.) We compared the performance of this system with one textbook method (a version of Salton's *tf.idf* method), one commercial indexing and retrieval system (SearchIt(TM)), and a series of simple systems that we implemented in order to explore different combinations of several traditional document retrieval techniques. These systems, called TWIDF, TWITF, IDF, ITF, NTWIDF, NTWITF, NIDF, and NITF, will be described below.

The experiments used the man page collection and the query set described above (a very challenging task for traditional document retrieval systems). We measured precision and recall at the cutoff point of ten retrieved documents, and we also computed a measure we call "success rate," which is the percentage of queries that have a successful answer in the top ten retrieved documents. The results are shown in Figure 8. I will discuss these results in some detail, but the principal observation is that Recall is dramatically better than all the rest, and this difference is statistically significant.

We chose to use the ten-document cutoff and the success rate measure for these experiments because internal studies of Sun's retrieval system users revealed that people normally do not look past the first ten hits in a result list. Also, for this kind of reference application, once a user has found an acceptable answer there is no need to find a second answer to the same question, so the presence of a single answer in a result list, rather than the number of answers for the same question, is what counts. Needless to say, this user profile is dramatically different from that of the hypothetical information researcher/analyst assumed by traditional document retrieval systems. This profile represents a distinct class of online information user, whose needs are significantly different from the customers traditionally served by information retrieval systems.

Retrieval Strategy	Success Rate	Recall	Precision
Recall	60.7	38.6	7.3
TWITF	47.6	28.4	7.4
SearchIt	44.0	28.5	7.4
NTWITF	36.9	24.7	5.2
TWIDF	28.6	17.2	4.8
Textbook tf.idf	28.6	14.8	2.9
NTWIDF	27.4	14.5	3.5
IDF	25.0	14.8	2.9
NITF	20.2	12.7	2.4
ITF	17.9	8.9	1.9
NIDF	16.7	9.2	1.8

Figure 8. The Recall system versus classical retrieval techniques.

We tested the performance of the Recall system against that of SearchIt, a commercial indexing and retrieval product developed at Sun Microsystems using a search engine from Fulcrum Technologies, a major supplier of commercial search and retrieval technology. We also compared it against a full-featured "tf.idf" algorithm, using log normalized counts and root-mean-square-weighted sums, taken directly from a standard information retrieval textbook [Salton, 1989] and implemented specifically for use in this experiment. The term "tf.idf" refers to a method that weights term occurrences in a document by the number of times they occur in the document, and normalizes them by the number of documents that they occur in. The name denotes the product of term frequency (tf), which is the number of occurrences of a term in a document, times the inverse document frequency (idf), which is 1 divided by the number of documents in the indexed material in which the term in question occurs. This basic equation characterizes a family of traditional retrieval methods that differ in detail, but share this basic strategy.

For the record, the man page collection consisted of 1819 files (one for each documented UNIX command), for a total of 10.65 megabytes of indexed source text. This is about the same size as the collections of abstracts on which most of the classical (pre TREC) document retrieval experiments were conducted. The man page collection resulted in a conceptual index (for the version of the Recall system tested) that occupied 19 megabytes. By comparison, the index of this same material produced by the commercial system SearchIt occupied 3 megabytes.

We compared the Recall system and the above two reference systems with a family of simple systems TWIDF, TWITF, IDF, ITF, NTWIDF, NTWITF, NIDF, and NITF, that represent all combinations of three common techniques used by various classical document relevance ranking methods. These systems were implemented to explore the relative merits of (1) normalizing the score of a document by the length of the document in order to compensate for the effects of differences in document size (abbreviated N for normalize), (2) weighting the match score of a term by the number of times it occurs in the document (ab-

abbreviated TW for term weighted) as opposed to simply counting one for each query term that occurs in the document, and (3) addressing the fact that some terms are better content-indicators than others by multiplying the weight assigned to each term by either the inverse of the number of documents in which the term occurs (abbreviated IDF for inverse document frequency) or by the inverse of the frequency of occurrence of the term in the collection (abbreviated ITF for inverse term frequency).

The systems in this family were named by the concatenation of the abbreviations of the features that they implement. For example, NTWITF (for "normalized, term-weighted, inverse term frequency") is a strategy that normalizes document scores by the length of the document (hence N), weights term matches by the number of times they occur in the document (hence TW), and normalizes the weight of a term by the inverse of the frequency of that term in the collection (hence ITF). The system named IDF does not normalize for document length and does not weight terms by the number of times they occur in the document, but does normalize each term weight by the inverse of the number of documents in the collection in which that term occurs (hence IDF). The system TWIDF is a simple version of the standard *tf.idf* family represented by the textbook algorithm. All of these systems are simple stereotypes of a corresponding family of systems that either have been or could be explored in the information retrieval community. They are stereotypes because they simply count the terms and sum them, with no fancy adjustments for log normalization or root-mean-squared averaging.

Conventional wisdom in the information retrieval community says to use inverse document frequency to normalize term weights in order to favor terms that are better discriminators among the documents in a collection. However, the rankings in Figure 8 suggest that (at least for this kind of query and this kind of material) normalizing by inverse term frequency is better. That is, the methods using ITF systematically outperform those using IDF except for the pure IDF case where neither term weighting nor document length normalization is used. It seems that the more important factor in determining the value of a term is its inverse term frequency (i.e., the inverse probability of the term's occurrence). This is correlated with inverse document frequency, but only approximately.

It is also conventionally argued that it is important to pay careful attention to details of the document relevance formula, such as using logarithms of some quantities instead of direct magnitudes and using root-mean-square weighted summations in certain cases. However, in this experiment, the performance of the full-featured *tf.idf* algorithm, using such refinements, is very nearly the same as the simple TWIDF stereotype (not quite as good, actually, in terms of precision and recall, but tied for success rate). Similarly, normalizing for document length is argued to be important when the size of documents in the collection may differ, but the use of this kind of normalization in the NTWIDF algorithm does not make a substantial difference in the success rate compared to the other two *tf.idf* methods. Despite the arguments that subtle differences in the way relevance scores are weighted and summed are important, it doesn't look like they make enough difference to counter the "family resemblance" that clusters these three *tf.idf* variations at comparable levels of performance in this experiment.

The fact that all three term-weighted inverse document frequency methods (TWIDF, textbook *tf.idf*, and NTWIDF) occur in a cluster in the success rankings of Figure 8 gives some reason to expect that other variations of TWIDF will perform similarly and that this posi-

tion in the rankings accurately represents the approximate level of performance that can be expected for any of the tf.idf family of methods. Similarly, I believe that the position of the TWITF method in the rankings represents the approximate performance level where modern "probabilistic" retrieval algorithms will cluster. This is because the equation for scoring document relevance for TWITF is essentially the equation for a Bayesian statistical estimate of the likelihood that choosing terms from a document would generate the terms of the query. This is the technique used by probabilistic retrieval systems to estimate the prior probability of relevance of a document to a query (before they start factoring in user-supplied relevance feedback to compute updated posterior probabilities).

Having now somewhat calibrated the performance ranking spectrum of Figure 8, we can see that the classical textbook retrieval methods are roughly in the middle, the commercial system SearchIt is in a cluster with the probabilistic retrieval techniques at the top of the range of traditional methods, and the Recall system is off the top of the scale. An exercise in statistical significance testing shows that Recall's position in the rankings is not an accident. For each comparison of Recall's performance with a classical retrieval strategy, Recall's improvement is statistically significant at better than a .03% level of confidence.

A key characteristic of the relaxation ranking retrieval algorithm is that it works well for small specific requests (two to four words) and small specific targets (passages ranging from a single phrase to several sentences). This is an area where traditional retrieval techniques tend to work poorly, as illustrated by their generally low numbers for this experiment. Since it is an area that is very important for many users of online information, the dynamic passage retrieval algorithm meets a significant need.

Another important characteristic of the relaxation ranking retrieval algorithm is that it seems to be especially good at exploiting morphological and semantic knowledge to handle paraphrase variations. An ablation study showed that the relaxation ranking retrieval algorithm in Recall still gave good performance even without the morphological and semantic knowledge normally provided by the conceptual index. When the Recall system was run without benefit of any morphological or semantic relationships, the success rate was 42.9, only slightly below the 44.0 of SearchIt, which contains a moderately good morphological variation capability. When Recall was run with morphology capabilities enabled, but no semantic relationships, its success rate was 52.4—well above SearchIt and well above the best probabilistic method (TWITF's 47.6). The relaxation ranking retrieval algorithm is clearly able to derive significant improvement from the use of semantic and morphological knowledge.

For comparison, we did some experiments using SearchIt on the same man page data and query set, with various subsets of the semantic knowledge used by the Recall system. We fed this knowledge into SearchIt's thesaurus capability. These experiments with SearchIt uniformly resulted in degradation rather than improvement, even when we hand fed the system only the facts that we knew would actually be useful in retrieving some correct answers. Essentially, the queries that were helped by these expansions were outnumbered by the queries that were hurt when spurious additional hits pushed correct hits outside the critical top ten window.

The contrast between the abilities of these two different methodologies to exploit morphological and thesaurus information was dramatic. It appears that this difference may stem

from fundamental limitations in the way vector-based and probabilistic document retrieval systems compute their scores (by summing items of positive evidence), compared to the way the relaxation ranking algorithm derives its penalty-based scores (by measuring deviation from an ideal target). The penalty-based scores assigned by the relaxation ranking algorithm appear to deal more effectively with thesaurus expansion (and also tend to produce noticeably good result rankings).

5.3 The user experience

Formal measurements of retrieval performance on a fixed query set are useful, but they are limited in their ability to fully assess the utility of a search system. In the case of the previous experiments, the formal measurements were even misleading, because they measured a collateral characteristic of the technology (the ability to retrieve documents) for no other reason than to compare the system with traditional systems. The true benefits of the technology for locating specific passages within the material, thus saving the user time and effort spent scanning and reading, were not being measured. Worse than not being measured, they were actually being penalized, because the Recall system, in response to the query "print a file," correctly located a passage in the man page for Sun's file manager that describes how to print files using the file manager's GUI controls. Unfortunately, the man page for the file manager had not been one of the "acceptable" answers selected in advance for this query by the human judge, presumably because printing files is not central to what the file manager does. It is even more interesting, therefore, that on this collateral task, for which the system was not designed, Recall still outperformed traditional document retrieval techniques.

To get conceptual indexing and retrieval technology into the hands of real users and see how they reacted to it, we implemented a third system, Solar. Solar was implemented as a network service with a Mosaic interface so that it could easily be made available to anyone on Sun's intranet. Some of the experiences of users with this capability are described in the next few subsections.

5.3.1 Indexed news articles

The first deployment of the Solar system was to automatically index Sun's subscription to the First! newsfeed from Individual, Inc., a service that provides selected news articles to customers based on a profile of interest. Sun's corporate library already made this available over Sun's intranet using a standard table-of-contents navigation interface. Solar was deployed to provide content searching over this same body of material. A modest user study was conducted with some of the first users of this system. After a pilot experiment with a librarian and a system administrator, a "natural habitat" study was run with four participants using the system in their daily activities. The format of the study involved:

- A 20-minute introduction, followed by a 4-day unsupervised trial
- Follow-up questionnaire
- Analysis of access logs

The principal results of this study were:

System performance was fine (response time, etc.)

Functionality positives:

Liked locating unexpected phrases ("quality" ==> "audio quality")

Searching was better than topic navigation (if specific terms)

Searching was more effective than concept outline browsing

Jumping to passage within document was good (if highlighted)

Showing context with search results was useful

Functionality negatives:

Repeated hits were a problem (some of the data had duplication)

Numerical relevance rankings needed to be understandable

No complaint about quality of concepts

Significantly, the design features of the system were consistently evaluated positively when mentioned, while the mentioned negatives were human interface issues that had not been addressed.

5.3.2 Indexed audio and video material

The first embedding of Solar technology in an application was its use to index the closed-caption text of recorded audiovisual material in the Indexed Media Streams (IMS) system developed at Sun Microsystems Laboratories (SunLabs) in Mountain View. The IMS system captures and stores broadcast television signals together with a time-synchronized record of their closed-caption transcripts, and makes them available for selective playback in a viewer, starting at any point in the stream. A conceptual index of the closed-caption transcript is used to provide navigation and search services so a person can move around in the audiovisual material. The use of a conceptual index and dynamic passage retrieval algorithm is especially well suited to this application because of the ability to locate specific passages in the material that can then be immediately viewed. This solves a significant problem associated with using recorded audio and video material, since without this ability, it takes considerable human time and effort spent scanning and probing with a fast forward button to locate specific items of interest in an audio or video stream.

A demonstration of this combined IMS/Solar system was on display at Sun's UltraSPARC™ product announcement on November 7, 1995. The demonstration provided a conceptual index to captured TV news broadcasts, so that someone could specify a topic of interest, get a response listing the phrases related to that topic that had occurred in the news, and then interactively view specific portions of the TV news broadcasts in which those phrases occurred.

This system was demonstrated live, to a live audience, by Duane Northcutt, the principal developer of the IMS system, and by SunLabs Director Bert Sutherland, as one of six presentations of Sun Microsystems Laboratories projects during the UltraSPARC launch activities.

As an illustrative example, in response to a request for "recent arson" in one week's indexed TV news, the best match found was:

RECENT (MARIN ARSONIST)
Morn.25.Oct@0-Caption.data

39 THE HUNT FOR THE MARIN
40 ARSONIST.
41 THIS MORNING.. THE SEARCH IS
42 FOCUSING ON NOVATO.. WHERE THE
43 MOST RECENT FIRES WERE SET.

and with a click of the mouse, the corresponding passage of video could be viewed on a Sun workstation.

The first line of the above display shows the phrases from the material that matched the corresponding phrases of the request—i.e., recent was matched exactly, and arson was matched by the phrase "Marin arsonist." This is useful information to an information seeker in deciding the significance of the match. The second line identifies the broadcast where the information was found. The rest of the display (with each line prefixed by a line number) is the passage determined by the dynamic passage retrieval algorithm. Notice that the terms "recent" and "Marin arsonist" occur in a different order in the passage than they do in the request and that they even occur in different sentences. The retrieval algorithm dynamically relaxed the search criteria of the original request until this passage was found. If a more exact match had been present in the news material, it would have been ranked ahead of this one and presented first.

5.3.3 Online documentation

An important area of application for conceptual indexing is software documentation. For multifeatured software products, a user frequently needs to consult the documentation as a reference. Unfortunately, software documentation is almost universally oriented around the names of the operators and operands involved, and not around descriptions of what they do or what you can do with them. It is easy to look up the name of a function in an index and find out what it does and how to use it. That doesn't help if you are trying to do something new and don't know the name of the operator that does what you want or even whether there is such an operator. An experienced programmer gets good at predicting what kinds of operators will be provided by a programming language or by a utilities library and may become good at guessing what they might be named. Less experienced users, or any user dealing with a new area is less able to cope. What you'd like to be able to do is somehow "look up" a description of what you want to do and have the index tell you the name of the operator. I refer to this as "content-based access." Conceptual indexing can make that possible.

One of my first personal uses of the conceptual indexing system was to index the online documentation for the text editor Emacs, a software program developed in the academic research community to edit a wide variety of different kinds of text files and program source code. Emacs has a powerful set of operators for doing a diverse range of things such as changing a selected region of text to uppercase or to initial caps or finding the place in another file where a selected word is defined. Emacs is famous/notorious for having an enormous set of commands, the most commonly used of which are invoked by combinations of control, meta, and/or shift keys with a single alphabetic character. It is difficult to become an Emacs user because of the high learning threshold that must be crossed before becoming

effective. Even experienced Emacs users don't know all of the commands. Emacs has a rich set of online documentation, but as expected, it is indexed by the name of the command, not the content describing what the command does. The following example illustrates the liberating experience of having a conceptual index to provide efficient content-based access to this material:

In order to experience this kind of content-based access, I indexed the Emacs online reference documentation, using the Pilot indexer, and kept the index active so I could refer to it whenever I had a question while using Emacs. At one point, I realized that I had forgotten the command that tells Emacs to move the cursor to the end of the file being edited. When I used the Pilot indexer to search for the request "jump to end of file," the Pilot system's automatic feedback from the conceptual taxonomy showed me that "jump" was a kind of "move." This did not automatically affect my request, since the retrieval algorithm automatically follows connections from query terms to more specific terms, but not more general ones. However, I recognized immediately that "move" (instead of "jump") would more accurately express what I was searching for. Consequently, I canceled the first request, and issued a new request for "move to end of file."

When this example was first encountered, I had indexed the online documentation for the Emacs commands and Emacs command keys, but not the tutorial documentation. In the material I had indexed, there were no hits that matched all of the elements of my request. However, the relaxation ranking algorithm automatically relaxed the requirements to tolerate a missing word and retrieved a set of ranked passages, the fifth of which was what I was looking for. It contained the phrase "move point to the end of the buffer" (i.e., using "buffer" where I had used "file"). This passage was ranked after four successive passages with the phrases "go to the end of the expression," "move point to end of current line," "goes to end of buffer if CHAR not found," and "go to the end of the expression" (same phrase as the first one, but in a different passage). Since the user interface in the Pilot system is set up to automatically go to the first hit, and provides a "next hit" button to step from one to the next, I found what I wanted after four mouse clicks, looking at the full hit passage in context each time. At any point, I could have asked for a ranked list of all the hits by merely pushing a different button, and I could then have gone immediately to any item on the list that looked promising by selecting it.

Four mouse clicks and a quick glance at each hit is not a high price to pay for finding this information, especially compared to the alternatives of looking for that information in a physical book or trying to guess the command by trial and error. However, a friend pointed out that there was also an Emacs tutorial document, and so I added that document to the collection and tried the query again with the resulting index. This time, the first hit contained the phrase "go to end of file" and provided exactly the information I was looking for. The wording was slightly different from, but subsumed by, my request and matched all of the elements of my query (with "go" subsumed by "move"). Needless to say, this hit was in the tutorial documentation, rather than in the command documentation searched previously.

This pattern of finding hits more successfully when the indexed material includes both tutorial and reference material repeated itself in another software documentation example, when we indexed the "man" pages for the Tcl scripting language. After we had constructed this index, a colleague came to us with the problem of trying to make some changes in a Tcl script that had been written by someone else. This person did not know Tcl and just

wanted to find out enough information to make the changes. What she wanted was an example of what I call "just-in-time learning"—the situation in which you suddenly need to know how to do something for a specific purpose and you don't have time to take a course. This sounded like a good opportunity to test out my hypothesis that conceptually-indexed material would support just-in-time learning.

Unfortunately, she didn't find what she needed for this particular task, but the experience illustrated another feature of the relaxation ranking retrieval algorithm—the ability to quickly conclude that what you are looking for is not in the material you are searching. In most information retrieval systems, when you don't succeed with your first request, you have to keep trying to see if some other way of wording your request might succeed. It is almost impossible to conclude that what you want is not actually there somewhere in some other terms that you have not managed to think of. With conceptual indexing and automatic relaxation ranking, however, if the best hit has a bad score and a few attempts to re-express the need all fail, then you can have some confidence that what you want isn't there. In this case, a few questions convinced us that the information required was not in the man page data we had indexed. She needed basic information that was probably present only in the Tcl book and was taken for granted by the man page authors. We subsequently indexed the Tcl book as well as the man pages, and when we indexed material about the Java language, we indexed two books and a number of discussion groups.

5.3.3.1 Efficient content-based access

To give you a more detailed picture of what querying is like with the kind of content-based access that this technology can provide, the following example shows a portion of the summarized hit list produced by the Pilot system for the query "move to end of file" using the Emacs tutorial documentation described above. In the listing, each hit entry is represented by a sequence number, a penalty score, a list of matching terms, an identification of the document in which the hit occurred, the positions of the hit passage within the document, and the actual text of the indicated passage. This information is displayed in the following format:

```
+++++ <hit sequence number>
(hit <penalty score> <list of matching terms>
 <file where hit was found> <beginning position> <end position>)
<retrieved text passage>
```

Example:

For the query: "move to end of file"

the first three entries of the resulting hit list are:

+++++ 1

(hit 0.115 ("GO" "TO" "END" "FILE")
"/home/wwoods/parser/emacs/emacs-tutorial" 5881 5898)

M-> Go to end of file

+++++ 2

(hit 0.15500000000000003 ("MOVES" "TO" "END" "FILE")
"/home/wwoods/parser/emacs/emacs-tutorial" 4984 5012)

which moves to the end of the file.

+++++ 3

(hit 2.8499999999999998 ("DASHES" (MISSING TO) "ENDS" "FILE")
"/home/wwoods/parser/emacs/emacs-tutorial" 15624 15753)

begins and ends with dashes, and contains the string "Emacs: TUTORIAL". Your copy of the Emacs tutorial is called "TUTORIAL". Whatever file you find, that file's name will appear in that precise spot.

The following excerpted portions of the corresponding texts illustrate a display of the respective hit passages in context (the passages are underlined, and the matching terms are bold). You will note that there is a gradual relaxation from good matches to successively less likely matches, with appropriate penalty scores to indicate the degree of poorness of the match. In this scoring system, penalty scores greater than 2 indicate substantial likelihood that a match is not useful.

hit 0.115 ("GO" "TO" "END" "FILE"):

M-a Move back to beginning of sentence
M-e Move forward to end of sentence

M-< Go to beginning of file
M-> **Go to end of file**

>> Try all of these commands now a few times for practice. Since the last two will take you away from this screen, you can come back here with M-v's and C-v's. These are the most often used commands.

hit 0.15500000000000003 ("MOVES" "TO" "END" "FILE"):

Two other simple cursor motion commands are:

M-< (Meta Less-than), which moves to the beginning of the file, and M-> (Meta Greater-than), which moves to the end of the file. You probably don't need to try them, since finding this spot again will be boring. On most terminals the "<" is above the comma and you must use the shift key to type it. On these terminals you must use the shift key to type M-< also; without the shift key, you would be typing M-comma.

hit 2.849999999999998 ("DASHES" (MISSING TO) "ENDS" "FILE"):

If you look near the bottom of the screen you will see a line that begins and ends with dashes, and contains the string "Emacs: TUTORIAL". Your copy of the Emacs tutorial is called "TUTORIAL". Whatever file you find, that file's name will appear in that precise spot.

Notice that the first two hits, with penalty scores near zero, are both correct answers to the original question, and that the third hit, which is not a correct answer, has a penalty score greater than 2. The fact that the third hit has penalty greater than two and is an incorrect result is a reliable indicator that none of the hits later in the ranking are likely to be correct. Not all queries do quite this well, but this illustrates the potential of the technology. Note that the system is not sensitive to how context determines senses of words, so it accepts "dashes" as a possible specialization of "move" even though in this context it is clearly a plural noun rather than a verb. In the first hit, "move" is correctly matched to the more specific term "go," while in the second, it correctly matches the inflected form "moves." It turns out that the frequency of erroneous words sense matches is not usually great enough to be a significant problem. Nevertheless, using contextual knowledge and world knowledge to resolve, or at least rank, the ambiguities of such words is an ongoing research topic as part of the conceptual indexing project at Sun, in order to discover techniques to deal with this problem in the cases where it matters.

5.3.4 Tools for working authors

As we have worked with this technology, it has become clear that some of the residual problems stem from problems in the original authorship, such as the fact that the Emacs command documentation speaks of a "buffer" without mentioning that what the buffer contains is the contents of a "file" (or the potential contents of a file if the buffer has not yet been saved). Without this key piece of information, matching "buffer" with "file" is no more likely than matching "buffer" with "character" or "bit." There is no strong a priori relationship between "buffer" and "file" that could be put into a taxonomy to make this connection. This tendency to leave important things implicit and unsaid is a fundamental characteristic of natural language that poses a serious barrier to automatically searching computer documentation. For example, in one version of the UNIX man pages, the man page for the command "lpr" (the command used to print files) never says that lpr prints files. It says, "lpr forwards printer jobs to a spooling area for subsequent printing as facilities become available."

Now you may think it's obvious that if a job is called a "printer job" and is sent somewhere for subsequent printing, then something is going to be printed, and if the next sentence says that (with various qualifiers) the job consists of copies of "each filename you specify," then maybe it will be files that are printed (although if you took it literally, you might expect to find file names in the printer output instead of the file contents). Computers don't find this kind of reasoning so obvious. It requires a lot of background knowledge and common sense to reason like this, and both of these are in short supply in the current state of machine intelligence. Often human readers of computer documentation have the same problems.

While conceptual indexing technology can't perform miracles, it has become increasingly clear that it can play a role (in fact several roles) in improving the quality of written documentation. The first level is simply to expose omissions of the kind we have just been discussing and feed those back to the documentation writer to be corrected. This is just a simple first step towards an intriguing idea of writing documentation with conceptual indexing as a target from the beginning. That is, the author of the material might start by using a conceptual taxonomy to systematically describe what system elements are being documented and what actions they are used for. Different ways of describing the same action that would not otherwise be taxonomically related could be separately expressed and explicitly cross referenced in the taxonomy by appropriate axioms. The description of how to do the described action could then be attached to the description of the action, using the structure of the conceptual taxonomy of actions as the organizing principle. This idea is discussed more fully in a later section on "taxonomic documentation."

Another way that conceptual indexing can help authorship is to help organize the raw materials that go into developing the documentation in the first place (e.g., system design specifications, focus group studies, e-mail messages, minutes of design meetings, etc.), so that the author can quickly find specific items of information when they are needed. A conceptual index of all the relevant input material can do a much more effective job than simply using folders of notes or a box of note cards, not only for software documentation, but for any kind of authorship. Also, conceptual indexing can be used to index the evolving product, so that the author can quickly check what has already been said, where it is said, how it is said, etc. The conceptual index can be used to quickly check whether terms have been defined before they are used, whether something presupposed by a passage has already been explained, etc.

We have exposed this technology to several working authors to evaluate whether the idea of using conceptual indexing to support authoring has merit. One of these was James Fallows, then Washington editor of the *Atlantic Monthly*, now editor of *U.S. News & World Report*, and a well-known author and radio commentator. To show him what the technology could do, we indexed a body of material that he had authored and let him poke around in it using the system. Despite the fact that the technology was still being developed and contained numerous bugs and omissions, Fallows concluded that the system could be very useful to him. In his published opinion of the technology, Fallows [1996] writes:

"I was less impressed by the failures than by how many sensible judgments the indexing system had made. Unlike computerized systems for translating one language into another, which often produce gibberish or ridiculous sentences, when the indexer failed it failed gracefully. It gave a number of clues for finding information, so that if one was misleading,

the others would get me there. At all times it let me observe and second-guess the structure of its logic."

6 Commentary

Conceptual indexing and retrieval using the relaxation ranking dynamic passage retrieval algorithm is a significantly different paradigm from that of traditional information retrieval systems, and one that appears to have significant ability to improve information access. The most salient difference, of course, is the focus on identifying specific passages of material in context, rather than just retrieving documents. However, experiments suggest that there is also a fundamentally important difference in the way that morphological and semantic subsumption knowledge is used and the way penalty scores are assigned by the relaxation ranking algorithm. The fact that this technology performs well for short queries, where the traditional paradigm is known to perform poorly, is one notable piece of evidence. It is also significant that morphological knowledge and semantic subsumption axioms produce significant improvements in the success rate of the dynamic passage retrieval algorithm, while adding the same knowledge to a state-of-the-art commercial search and retrieval system caused degradation. In the next subsection, I will look more closely at the characteristics of the penalty-based scores assigned by the relaxation ranking algorithm and discuss some effects of this method on ranking. In later subsections, I will discuss strategies for how best to express queries using this new paradigm, and some ways to apply the technology.

6.1 Penalty-based scoring

Almost all existing retrieval systems (other than simple Boolean retrieval systems that do not assign relevance rankings) assign scores to documents based on accumulating evidence in favor of the document derived from counting occurrences of words in the document. There is a fixed and recognizable lower limit of zero for the score of a totally irrelevant document, although generally a document can be irrelevant even with a significantly nonzero score. Depending on how scores are normalized, there may or may not be an upper limit on the score assigned, but even for probabilistic methods where this upper limit is 1, relevant documents rarely come anywhere near that limit. Thus, there is no way of determining in isolation whether the score assigned to a document by these methods is a good score or a bad one. It is only the relative value of these scores, within the context of a particular collection of documents, that determines a ranking of documents in which hopefully the better ones are ranked ahead of less good ones. Consequently, the actual scores are not meaningful, and in many commercial systems are hidden from the user, or replaced by simple indicators of qualitative levels of goodness, or mapped arbitrarily onto a scale from 0 to 100 because that is a scale that users will recognize as familiar.

In contrast, the penalty-based scores assigned by the relaxation ranking method provide a distinct meaningful score for a perfect match (i.e., zero penalty), and many retrieved passages achieve this score. Moreover, the magnitudes of the assigned penalties have an explicit calibration that can be meaningfully interpreted in isolation. For example, in the Pilot system, a 10 point penalty corresponds to a missing significant word, a 5 point penalty corresponds to a missing less-important word, and a penalty in excess of 2 points indicates that the terms are widely separated in the text and/or out of order and so are potentially less likely to be related in the way that the query requires. Penalties smaller than 1 tend to be minor

adjustments that assign a preference order among hits that are all likely to be relevant to some degree. Consequently, for these systems, the magnitude of the penalty score can be meaningful and informative, and therefore useful to the information seeker.

Another important characteristic of the penalty-based scores assigned by relaxation ranking is that, since the scores are meaningful in isolation, result lists from searches for the same query on different collections can be simply collated together on the basis of the penalty scores to produce a suitably ranked aggregate list of results. This is in contrast to traditional scoring methods that are normalized by statistics of the collection. For these systems, scores assigned to documents from different collections are not meaningfully comparable, and simply collating results lists together based on their scores may produce an aggregate ranking that places good hits from one collection after bad hits from another.

6.2 Effective search strategies

Because conceptual indexing retrieval is a new paradigm, one might expect to learn correspondingly new ways to use it most effectively. For example, the traditional document retrieval paradigm has long been noted for performing poorly on short queries. Researchers have repeatedly sought ways to encourage users to provide longer queries (so that word counting statistics have more chance to succeed). Indeed, *relevance feedback*, one of the most successful innovations in information retrieval research in the last 30 years, can be seen as a way to get the user to provide a longer query. Relevance feedback is the technique whereby the user tells the system which documents from an initial retrieved set are good and which ones are not, so that system can take that information into account and try again. In the second round of a relevance feedback search, the query is effectively the collection of all the terms that occur frequently in the good documents and do not occur frequently in the bad documents. (A simpler version of this technique simply takes a relevant document as a single, big, new query.) Hence, the most effective use of these systems is to find (somehow) an initial query that will recover at least one good result and then use relevance feedback to get the system to find more like it (or them). This style of use makes sense for a researcher looking for comprehensive references, but is not an effective style for the typical online user who simply wants a quick answer to a question.

The typical users of today's online information services are not willing to provide a long query, and often, once they find a single acceptable answer, their need is met, so relevance feedback doesn't help them at all. The important task for them is to find the first acceptable answer with a minimum of hassle. The conceptual indexing paradigm supports the needs of these users. In the rest of this section, I will present some things I have observed about how to phrase requests to get the most out of this technology. This theme is covered more fully in a separate document, *Guidelines for Precision Content Retrieval* [Woods, 1995a], which was prepared as a guide for users of the system.

In order to use conceptual indexing technology most effectively, it is useful to remember how relaxation ranking scores are derived. Specifically, these rankings are derived from a combination of how many of the terms of the query are represented in the passage, what forms or synonyms of those terms occur or what other related terms occur in the passage, and how the matching terms are related in the passage (e.g., are they adjacent and in the same order, are they in the correct syntactic relationship, how closely do they occur together, etc.). The ideal passage is one that is an exact replica of the query. This standard is progressively relaxed (with the introduction of corresponding penalties) to allow other in-

flected forms of the words, more specific terms, intervening words, different word orders, intervening sentence boundaries, and some terms missing. Thus, an ideal query is a phrase that is likely to occur literally in the passages sought (and unlikely to occur in irrelevant passages). The relaxation ranking algorithm will automatically work from that to find passages that depart from this ideal.

Often there is a single word that is a very good content indicator for passages of interest and is not likely to occur in passages in which you are not interested. If so, then that term is an ideal query all by itself. For many information needs, such a term, used alone and matched exactly, is all that is required. In other cases, however, some desired matches will make use of different forms of that term (e.g., "arsonist" for "arson") or will contain more specific terms than the term requested (e.g., "abduction" for "kidnapping"). The conceptual indexing retrieval system will automatically search for different inflected forms of the terms you specify and also for more specific terms and their inflected forms. Thus, to be most effective, your query should consist of the most general term or terms that you think will describe what you are interested in, without being so general as to retrieve things you don't want. Feedback from the conceptual taxonomy can give you a good idea where the term you choose fits with respect to other terms and whether a more general or more specific term would be better.

The system treats inflected and derived forms of a word (e.g., plurals of nouns and derivations of words such as "reduction" from "reduce" + "tion") as if they were more specific than the base form of the word. Consequently, your query will be more effective if you use the base form (singular form for a noun, root form for a verb) unless you are interested only in a particular inflected form. Thus, for example, "shooting" is a better query than "shootings" (even though the latter may feel more natural in some cases), because it will automatically match both "shooting" and "shootings" (the latter will only match "shootings"). A still more general query would use the base form "shoot," which will match "shoots" and "shot" as well as "shooting" and "shootings." Notice that the system knows irregular forms like "shot" as well as regular forms that can be derived by rule. (This paragraph describes the behavior of the Solar system; in the Pilot system, all inflected forms of a word are automatically matched, regardless of the inflection of the query term, but matches with the same inflected form as the query are preferred.)

When a single word is not sufficient to specify exactly what is sought, usually only two or three words are necessary to form a suitable query. The relaxation ranking retrieval algorithm is especially good at handling such queries, and in fact runs faster for shorter queries, so your query should be just long enough to cover the important desired concepts. An ideal query is a two- or three-word phrase that is likely to occur in the passages of interest in exactly the order specified in the query. Using uninflected forms of the words in a query is usually best, since these automatically subsume all of the inflected forms. Thus, "school bus accident" will be a more effective query than simply "accident" if it is only school bus accidents that you are interested in.

In many (in fact most) information retrieval systems that claim to allow what they call "natural language" queries, the "natural language" input is interpreted by the system simply as a collection of words to look for. This sometimes leads users to adopt an unnatural query style that consists of a simple list of alternative words of interest. However, this style is not good for the relaxation ranking dynamic passage retrieval algorithm, because this system

will try to find a passage where all of those words occur rather than interpreting them as alternatives. For this technology, it is better to find general terms in the conceptual taxonomy that cover the ranges of alternatives in which you are interested. You should pose a query as a sequence of terms each of which should be represented, if possible, in a passage of interest, and preferably in that order. This gives you more control and usually with less effort than simply mixing required terms and alternative terms indiscriminately in a single list.

6.2.1 Refining a search

Occasionally when examining the list of results from a search you will find that it includes passages you are not interested in because your chosen query was inadvertently too broad. If the number of such distractors is not large, you can simply skip over them by eye. However, if there are many distractors, you might want to refine your query to eliminate some of them. This process is often called "narrowing" in the information retrieval community. For example, in one search of the TV video news for "school bus accident," looking for updates on a nationally-publicized accident that had happened in Illinois where a train ran into a school bus, the first ranked hit referred to a school bus accident that occurred in San Francisco, where a school bus ran into another vehicle. If you want to focus more on just the Illinois accident, and want more items than you have already seen, then you can rephrase the query, adding the modifier "train" or Illinois (e.g., "train school bus accident," or "Illinois school bus accident", whichever is most likely to express your range of interest). If neither of these is sufficient alone, you could ask for "Illinois train school bus accident," but this is worth doing only if neither of the shorter queries eliminates enough distractors.

You will occasionally get matches to query terms that are accidental, as in the following match to "train school bus accident" that picked up on a reference to "training" (which is clearly not related to the sense of "train" that was intended by the query term). Generally, our experience has been that the frequency of such false hits is small. (In this case, it actually picked up a reference to the correct incident, but for a wrong reason.)

```
134 >>> THAT ACCIDENT NEAR CHICAGO
135 MAKES THE REST OF US THINK
136 TWICE ABOUT THE SAFETY OF KIDS
137 IN SCHOOL BUSES EVERYWHERE.
138 WHAT KIND OF TRAINING DO
139 SCHOOL BUS DRIVERS RECEIVE
```

6.3 Cascaded indexing and retrieval

Conceptual indexing technology was designed to help people find their way to specific information within a body of material. This is useful even when the body of material is a single document or a small collection of related documents. In particular, it is useful when the set of documents is the result of a search in a traditional document retrieval system. Traditional document retrieval systems, after they have done their job, leave their users with the nontrivial task of reading the resulting documents to see if the information required is actually there. Conceptual indexing technology can pick up at this point and make it possible for the information seeker to conceptually access the content of the retrieved docu-

ments to find specific items of information quickly. In this way, conceptual indexing can be viewed as complementing document retrieval technology rather than competing with it. The two technologies actually address substantially different problems.

Cascaded indexing, a two-stage process consisting of a traditional search in high recall mode (and therefore tolerating low precision) followed by a conceptual index of the results, looks extremely attractive for many application areas. I have tried this kind of cascaded indexing by taking the output of traditional searches done by the online retrieval systems used in Sun's research library and feeding the results through the conceptual indexer before attempting to read them. I can then examine the content of this material much more effectively by browsing the conceptual taxonomy to see how terminology is used in the material, and using the dynamic passage retrieval technique and links from the taxonomy, to move around in the material. The conceptual index provides a powerful tool for mining the resulting material.

Cascaded indexing can be an important element for all kinds of analysis of material retrieved by conventional search. This is especially true for analyzing the materials that result from searching the very large collections available today (such as the World Wide Web). As larger collections of material have become available, it has been discovered that some of the classical retrieval techniques (which were mostly developed for collections of abstracts) have limitations that were not previously apparent. Figure 9 lists some of the unique problems associated with indexing and searching very large collections.

- Users can be overwhelmed by the number of results
- Eliminating duplicates is necessary
- Ranking becomes critical
- Precision becomes critical
- Document size variations can adversely affect scoring
- Narrowing techniques are essential
- Advanced techniques are often sacrificed for speed

Figure 9. Unique problems of large collections.

It is not uncommon for a simple search of the World Wide Web to produce a results list with 20,000 hits, overwhelming the user with the task of making sense of the result. Typically, Internet Web searchers provide the user with the first 10 hits and continue to provide additional blocks of 10 until the user finds something acceptable or gives up. If the user has a simple information need and the answer shows up in the first 10 or 20 hits, then this is not unreasonable. However, if the user has serious research interest in the results, then it may be important to know what's in the rest of those hits. Using a conceptual index of the resulting hits, can make it possible for an analyst/researcher to effectively mine such an abundance of potentially relevant information. Without a second stage of something like conceptual indexing, it is hard to see how one can make much use of 20,000 hits in response to a query.

6.4 Distributed indexing and retrieval

Another problem that arises when one attempts to index the World Wide Web is that the information on the Web keeps growing and changing faster than Web crawlers and monolithic global indexes can keep up with it. Eventually the approach of having a single index of the whole Web will become impossible as the Web continues its exponential growth. An approach that is much more likely to grow with the Web is to distribute the indexing task among many indexes on the Web, and distribute the search for information requested in the same way. One then needs a way to integrate the results from such a distributed search to give the user a coherent response. A major problem in integrating such results is that, unless the document evaluations assigned by the different indexes are commensurate, a strategy of (say) taking the top 100 hits from each site, merging them, and pruning the result to obtain the 100 best hits on the Web, will be likely to prune away many hits that are better than some other hits that it keeps.

As noted above, traditional retrieval systems (including probabilistic retrieval systems) that normalize document scores using statistics of the indexed collection, tend to produce document scores that are not commensurate across collections. On the other hand, the penalty-based scores assigned by the relaxation-ranking algorithm are automatically commensurate and pose no problem to the strategy of merging and pruning results. Thus, conceptual indexes can be easily decomposed and distributed arbitrarily, and the results of conceptual index searches can be merged directly with no commensurability problems.

6.5 Size of indexes

A topic of interest to potential implementors of this technology is the size of a conceptual index and how conceptual indexes scale with the size of the body of material indexed. We don't have a definitive answer to this question at this time, since the experimental systems we have been running have not been set up to either optimize or measure the growth rate of index sizes. However, we have developed some techniques for keeping the size of the index manageable. Currently, indexes exist in one of two modes: a working index that includes approximately 50% empty space in order to optimize the efficiency of additions, and a compacted index in which the empty space has been squeezed out. In one experiment, compacted indexes required approximately the same space as the material being indexed, and working indexes used approximately twice as much space. This is in the range of sizes of indexes of conventional retrieval systems, which are typically 1-3 times the size of the material indexed when word positions are recorded and no compression of the index is attempted. Compacted indexes can be used for any body of material that remains fixed. Compaction can also be applied at any time to save space. If a compacted index is used for further additions, extra space is automatically inserted, and the size of the index gradually increases. This suggests that the index size of a working index may range between 1 and 2 times the size of the indexed material.

Traditional indexes can be as small as 50% of the size of the indexed material when compression techniques are used. This figure assumes that the positions of indexed terms are being recorded in the index. If word positions are not recorded or if words from chosen "stop lists" are not indexed, then even smaller indexes can be achieved (with compression). Similar techniques can be applied to conceptual indexes, although they may or may not have comparable compression factors. One can expect the index of term occurrences for a

conceptual index to take the same space as comparable indexes for traditional search and retrieval technology. However, the space required for the conceptual taxonomy of phrase concepts is less predictable. Experience on a variety of collections has shown that, while the growth rate in the number of distinct words in an index tends to level off after a reasonable number (typically 20,000-40,000) words have been indexed, the number of distinct concept phrases continues to grow. At least, that has been the case for the collections we have indexed so far (up to approximately 200 megabytes of text). This growth may eventually slow, like the growth of individual words does, but may level off with a much higher number of phrases in much larger collections. As a sample data point, the SunExpress catalog cited earlier contained slightly over 20,000 distinct words and a total of 100,000 concept phrases for a collection of approximately 1 million words. Based simply on the ratio of phrases to words, one might guess that a conceptual index might be as large as five times the size of a conventional index, before any compression tricks are applied. However, although the number of phrasal concepts is large, the individual phrases don't occur as often, so the space required for recording positions is not as great as for individual words. Moreover, an indexed phrase can be used to avoid the need to separately index the words that occur within it (by later following links from words to concept phrases and then to positions). Using this technique, one might achieve conceptual indexes that are actually smaller than equivalent keyword indexes.

So far, we have applied conceptual indexing technology to relatively small collections by today's standards (although many of them involve more text than the collections used for most of the classic information retrieval research experiments). The largest conceptual index we have constructed had something over 1 million concepts and was still a relatively modest body of material by today's standards. We have yet to determine how large a collection we are able to handle with a single index using this technology. The theoretical characteristics of the algorithms are favorable to indexing large collections, since the classification algorithms demonstrate performance that is logarithmic in the size of the indexed material. The principal item of uncertainty at this time has to do with the efficiency of secondary storage access when indexes become too large for a good working set to be kept in a memory cache. However, the distributed indexing techniques described in the previous section should enable the technology to handle arbitrarily large collections efficiently (effectively providing a larger cache as well as more processing power).

For the moment, our goals are to be able to index collections on the order of several hundred megabytes in a single index. This will be sufficient for most online documentation collections, for creating indexes to publications such as CD-ROM encyclopedias and online product catalogs, and for indexing the output from traditional search engines in cascaded indexing applications. Indexes to much larger collections can be obtained by using distributed indexing techniques, and much smaller indexes can be obtained by storing compressed traditional indexes and reindexing the results of traditional search algorithms, using cascaded indexing.

6.6 Semantic ambiguity

In an earlier section, it was pointed out that many English words are ambiguous and the system needs to take this into account. It would be nice if the lexical analyzer were able to resolve semantic ambiguity of tokens as it encountered them, determining which sense of a word was being used at each point in the source material. Linguists call this process "se-

mantic disambiguation," a technical term for "resolving ambiguity." Semantic disambiguation is difficult for computers, since the knowledge human readers apply to selecting appropriate senses depends on the context of occurrence in subtle ways that cannot yet be emulated on a machine. Even human readers are sometimes left uncertain as to the intended sense of a word in context. In fact, the distinction of word senses is a very subtle art. Dictionary definitions often distinguish senses that ordinary people find difficult to separate, and ordinary people don't generally make such distinctions well or in much detail.

On the other hand, people easily notice when there is a gross mismatch between the intended sense of a request and other senses that are not relevant. In the cases that matter for information retrieval, the difference is usually dramatic. For example, a hit on the word "dash" for the query concept "move" is a clear mismatch when the occurrence refers to a punctuation mark. Unfortunately, determining that the occurrence does refer to punctuation mark involves understanding the sentence in which the word occurs and choosing the interpretation that makes the most sense. This is more than today's natural language understanding systems can accomplish on more than a limited domain of discourse.

For a computer to disambiguate word senses the way a person does would involve analyzing the entire sentence, including all the possible interpretations of each word and then searching among the possible ways to interpret the sentence for the one that is most preferable, using such knowledge as what kinds of verbs apply to what kinds of objects, what things are likely or unlikely to happen, and what people are likely to say. This requires a great deal more knowledge than today's computer language understanding systems possess, and requires considerable processing power. Moreover, mistakes in this decision would lead to failure to make connections between a user's query and potentially relevant passages, so if the system is going to eliminate senses, it should be pretty sure it's right.

In the Pilot system, we have taken an approach that attempts to make use of linguistic and general world knowledge, while still recognizing that the system doesn't begin to know as much as a human reader and is not nearly as skilled at word sense disambiguation. This strategy involves distinguishing three places where semantic ambiguity can enter the picture, and handling the problem differently in each place: in the request, in the indexed material, and in the intervening conceptual taxonomy.

1. In the request, the user is typically available at the terminal and can be asked to resolve ambiguities if necessary. However, most users do not want to be hassled with an interface that forces them to be specific about the senses of the words they intend, even to the extent of choosing from an automatically provided menu of possible senses. (One of the reasons natural languages have semantic ambiguity is because it is not efficient for people to be that specific when they communicate.) Our strategy is to automatically consider all possible senses of a query term and answer the request accordingly. Typically, the proximity of other terms in a request will tend to favor the correct senses, and the user will have to resort to specific semantic disambiguation only if there are too many distractors (and with relaxation ranking, there typically aren't).

2. In the indexed material, there is no available author to ask, but one does have a lot more context than for a short query. Various statistical techniques can be used to determine the most likely senses of words in such contexts, assuming one has a good statistical model of the language used in a domain. These models have a certain error rate. They can

generally get the right sense of a word whose senses are different in different subject domains, but can't distinguish different senses that can be used together in the same domain. They can't, for example, distinguish the two senses of "interest" in: "It was not in his interest to pay more interest." Another technique, called part-of-speech tagging, attempts to disambiguate words as far as syntactic part-of-speech is concerned, again typically using a statistical model. Our experience is that part-of-speech tagging alone is counterproductive in information retrieval applications, since when a word is ambiguous with respect to part of speech, both senses are often semantically related to the same concept, and hits on either sense are good hits—e.g. "change" as a noun or "change" as a verb are both good hits for a query about change.

The approach used in the Pilot system has been to treat each word occurrence as ambiguous with respect to all of its senses and to use the taxonomy to connect that word to all its senses. This means that one may find hits that use the wrong sense of a word, but it guarantees that nothing will be missed through errors in sense disambiguation. We rely instead on relaxation ranking to order the hits so that the most likely ones occur earlier most of the time and expect to use query refinement techniques for dealing with the occasional cases where the number of spurious hits is excessive.

3. In the conceptual taxonomy, it is important to make word sense distinctions, because as one follows paths through the taxonomy to make connections between concepts, shifts in word sense along the way can lead to bizarre results. For example, a taxonomy might record that "average" is a kind of "mean" and that "mean" is a kind of "cruel" but we would not want to conclude that "average" is a kind of "cruel." I call this the "equivocation" problem, since it arises from a shift from one meaning of a term to another in the middle of a chain of inference. The potential problem is well illustrated by an example from an early version of the Pilot system, in which word sense distinctions were not made. On one occasion, the concept "book technology" was classified as a kind of "document frequency," because a book is a kind of document and the subsumption algorithm found the chain of locally valid relationships between "technology" and "frequency" shown in Figure 10 (where the local relationship in question is some-sense-of-subsumes-some-sense-of).

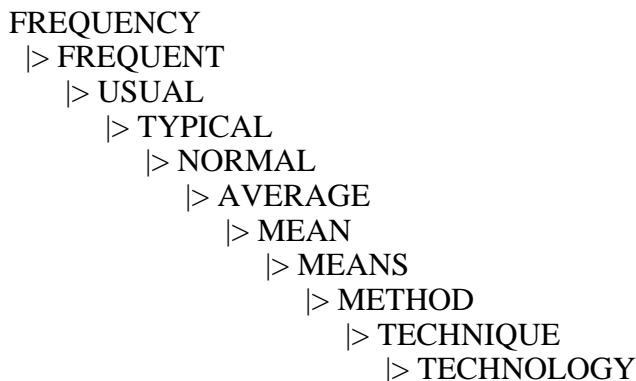


Figure 10. A chain of false "subsumptions" containing "equivocations" (a problem when word senses are not distinguished in the taxonomy).

The problem is that the relation some-sense-of-subsumes-some-sense-of is not transitive, because it is possible to shift the sense in the middle of such chains. Problems of this magnitude are extremely rare, but this example illustrates the nature of the problem and how subsumption testing can sometimes get bizarre results if word sense distinctions are not made.

In the Pilot system, the equivocation problem was dealt with by using "pseudowords" to name specific senses of a term when specifying subsumption axioms, so that, when necessary, these axioms are stated between named senses of words rather than potentially ambiguous words. Pseudowords are artificial terms with a formal structure that is meaningful both to the machine and to a human reader and behave as terms that denote specific senses of words. In the Pilot system, the pseudowords are designed with a syntax that is efficient for the machine to recognize and analyze. They start with a distinguished symbol (!) which is not normally used to begin a word, so that algorithms can recognize the presence of a pseudoword by a simple check on the first character of the name. This is followed by an explicit syntactic category (n, v, adj, ...) to indicate the syntactic category of the intended sense, which is separated by a slash (/) from the word of which this term is a sense. This is optionally followed by another slash and a disambiguating tail that mnemonically distinguishes this sense from other senses of the same word (when syntactic category alone is insufficient). Usually, the disambiguating tail is a more general concept that distinguishes this sense from others. For example, two senses of the word "standard:" !n/standard/flag and !n/standard/criterion, are distinguished by the indication that the former is a kind of flag, while the latter is a kind of criterion.

Pseudowords can use syntactic categories nm and nc to indicate mass noun and count noun senses of a word (e.g., !nm/injury vs. !nc/injury) and vt and vi to indicate transitive and intransitive senses of verbs (e.g., !vt/change vs. !vi/change). Pseudowords in the Pilot system also provide a syntax for naming specific inflected forms of words by including annotations such as +s, +past, +3sg, +ing, etc. after the word. Pseudowords can be used whenever a specific sense of a word needs to be referred to. In addition to their use by taxonomy authors to express subsumption relationships, they can also be used in queries to express specific intent. In principle they could be used by authors of source material, which could then be displayed in ordinary form by automatically suppressing the syntactic annotation and disambiguating tail. The internal form of pseudowords is designed to be easy to type and easy for a lexical analyzer to recognize efficiently, but for display to ordinary users, they can be transformed into a more intuitive bracket notation as indicated below:

!adj/standard	[adj:standard]
!n/standard/flag	[n:standard/flag]
!n/standard/criterion	[n:standard/criterion]

Senses of a word are considered more specific than the word itself (i.e., are subsumed by the word itself), so that requests for a word will automatically subsume any of its senses. Requests for specific word senses can be made which will only subsume concepts subsumed by the indicated senses, if the user is willing to specify the specific sense desired. Thus, only at the source text level is there a residual ambiguity where a specific sense of a concept has to be matched against a term in the source material whose intended sense is unknown. In practice, this level of ambiguity is generally not onerous for queries of two or three words and using the relaxation ranking passage retrieval algorithm. Nevertheless,

we have been exploring techniques to attempt to rate different senses of words in context.

6.7 Comparison with related work

The body of work presented here involves a combination of techniques from a number of different areas of endeavor. Thus, when comparing this work with other work, it is necessary to compare different parts of the system to the corresponding bodies of work with which they share goals. To my knowledge, there is nothing that is directly comparable to this overall system.

In the next section, I will compare the knowledge representation aspects of this system with other knowledge representation work, and in subsequent sections will take up the information retrieval aspects and the language processing aspects.

6.7.1 Related work in knowledge representation

The taxonomic conceptual structures used here belong in a family of systems whose progenitor was the KL-One knowledge representation system developed at BBN [Brachman & Schmolze, 1985]. Woods & Schmolze [1992] give a survey of this family of systems. However the system used here is not quite the same as most of the other systems in the KL-One family. The principal difference lies in the criterion of structural subsumption and in the use of role tags. Woods [1991] presents an argument for an approach based more on the "intensional structure" of concepts rather than their extensional semantics, and that approach has been used here. Woods [1991] also introduces the notion of quantificational tags used to express the quantificational import of roles and describes a number of such tags and their influence on the criterion of structural subsumption. The ME ("some") tag used here comes from that work.

Woods & Schmolze [1992] present an integrating framework in which various KL-One-like systems can be described and compared. In the terminology of that survey, the conceptual indexing system uses the "c-some" role operator. The c-some operator expresses the constraint that some value of the specified role satisfies the specified constraint. This relationship is the same as that expressed by the quantificational tag ME in Woods [1991] and represented here with the tag "some." In contrast, most KL-One-like systems focus more on the role operator "all," expressing the constraint that all values of the specified role satisfy the specified constraint. Hence, the descriptions here deal primarily with requirements that there exists some value of a role satisfying a value description, while most members of the KL-One family primarily use restrictions that all values of a role must satisfy. The c-some operator is a relatively late development in the KL-One family, and is not available in some KL-One-like systems. We have found that the c-some operator correctly expresses the semantics of normal topic phrases.

As mentioned, the system presented here primarily addresses a structural relationship among conceptual descriptions (of the special kind that Woods [1991] calls "*intensional subsumption*"). Much of the KL-One research community, however, has sought a definition of subsumption that is based on *extensions* of concepts, without reference to the structure of the conceptual descriptions. Basing subsumption on an extensional criterion has led to a number of unattractive complexity results in the KL-One research community. Consequently, since the theoretically-justified behavior of these systems is intractable, most ac-

tual KL-One-like implementations have adopted more efficient structure-based subsumption algorithms as a matter of expedience. Woods [1991] argues that subsumption criteria should be intensional rather than extensional, and provides a foundation for an intensional semantics for subsumption that admits efficient algorithms.

The system presented here also differs from KL-One in its handling of individual concepts. Neither KL-One nor any of its successors allow an individual concept to be further specialized in the way that is provided for here. The ability to further specialize individual descriptions allows a system to capture knowledge about incompletely described individuals and relate them to other individual descriptions that may be more complete or incomplete in different ways.

The conceptual structures used here are also similar to Sowa's conceptual structures [Sowa, 1983]. Sowa does not distinguish quantificational tags on roles, but instead takes all roles to have the semantics expressed by the ME ("some") role tag used here (i.e., expressing the c-some relation, in the Woods and Schmolze terminology). In that respect, his treatment of roles is similar to that used here. However, Sowa does not provide for specialization and generalization of role relations (only allowing this for the role values). In contrast, we have exploited the ability for the subsumption criteria of Woods [1991] to generalize and specialize role relationships (which we referred to as "relational abstraction") to capture some important elements of linguistic ambiguity and vagueness. Like most of the systems in the KL-One family, Sowa also follows the traditional practice of first-order logic by treating individuals as atomic entities and not allowing them to have decompositional structure or to be further specialized. Thus, his system, like these others, cannot address the phenomena of different levels of generality in the names of individuals.

The conceptual indexer presented here makes use of a simplified range of conceptual description, both linguistically and conceptually, compared to the capabilities of most knowledge representation systems. In particular, we have restricted attention to a delimited subset of what I called "type 1 descriptions" [Woods, 1991]. Roughly, type 1 descriptions are those descriptions that can be expressed in English without the use of pronouns, variables, or other expressions of *coreference*. Here, we have further limited ourselves to a few specific noun phrase patterns and verb phrase patterns that, for example, contain no relative clauses, no possessives, and no complex nominalization phenomena. Nevertheless, this class of descriptions has proven to be quite effective for indexing material and searching for concepts.

Another difference between this technology and that of other KL-One-like knowledge representation systems is scale. Most systems in the KL-One family are limited to relatively small knowledge bases and are happy to be able to handle a few thousand concepts. In contrast, using the Solar system, we have routinely constructed much larger taxonomies (the largest being over a million concepts). Also, most other systems in this family are general purpose systems for representing knowledge and drawing inferences, and are not integrated with the natural language processing capabilities necessary to automatically extract and analyze concept phrases from unrestricted text.

6.7.2 Related work in information retrieval

Most existing information retrieval systems are document retrieval systems—that is, their job is to determine which documents in a collection are relevant to a user's request. They

stop short of providing help in finding the information within those documents. Only a few systems attempt what has come to be called "passage retrieval," and in most such systems, the passage granularity (e.g., sentence level, paragraph level, or page level) is chosen at indexing time and these units are then indexed as if they were small documents. Sometimes, individual sentences are indexed and then combined together at retrieval time to produce passages. (See Salton et al. [1993] for examples.) Some of these, like the Recall system experiments, use the passage retrieval concept only to assign a score to a document as a whole. In contrast, the work presented here focuses on identifying the locations within documents where the requested information is likely to be found and constructs passages at retrieval time from the positional information associated with indexed words and phrases to determine the smallest passage size that satisfies all (or as many as possible) of the query elements. Moreover, most other passage retrieval systems use the same scoring methodology used by vector-based retrieval systems (i.e., they accumulate elements of positive evidence). None of them use the kind of penalty-based relaxation ranking scores used by the dynamic passage retrieval algorithm presented here.

The system presented here addresses many problems that have traditionally been addressed in the information retrieval community by concepts of synonymy. That is, the combination of conceptual indexing with relaxation ranking dynamic passage retrieval addresses the ability to relate similar phrases expressed in different words. Most information retrieval systems that address this issue do so with a thesaurus of word synonyms, sometimes replacing the original words with selected "canonical forms" at the time of indexing (unfortunately introducing noise and loss of information into the index). We have already discussed the limitations of automatically expanding a query with synonym thesauri. Similarly, many document retrieval systems have capabilities for morphological or hierarchical thesaurus expansion. They do not, however, impose a penalty on such cases in order to favor hits that use the exact query terms as opposed to the hierarchically-expanded terms. The results of our experiments show that the assignment of such penalties by the relaxation-ranking method can make a big difference in retrieval performance and in the ability to derive benefit from morphological and semantic knowledge.

6.7.3 Related work in morphology

Much of the current morphological work in computational linguistics centers around the use of a morphological paradigm known as "Kimmo systems" or "two-level morphology" [Koskenniemi, 1983]. These systems use finite-state transducers to represent the bidirectional transformations between roots of words and their derived and inflected forms. These systems compile sets of prefixes and suffixes and a vocabulary of known roots words with syntactic category information into transition automata that can map surface word forms into a sequence of its morphological elements (zero or more prefixes, followed by a root, followed by zero or more derivational suffixes, followed by an optional inflectional suffix). At least one such system parses the resulting sequence of elements to determine the possible sequence of application of prefixes and suffixes and corresponding syntactic category of the result. This system produces all possible analyses that are consistent with the rules of affix composition. When a word has more than one analysis, there is no mechanism to prefer one interpretation over the others. Kimmo systems are merely one of many approaches to morphology. Sproat [1992] provides more information about morphology and approaches to morphological analysis.

A basic Kimmo system provides only the rules that generate the sequence of morphological elements from the surface form. It does not specify how these elements combine to form an interpretation of the whole. Some systems contain an additional parsing stage to figure out how the sequence of elements could be assembled to derive a word. In principle, the automata of the Kimmo systems should be able to operate in either direction—i.e., analyzing or producing surface forms. In practice, however, the rules in these systems tend to be written with one direction in mind, and rules written from one perspective often don't work correctly in the other direction. Also, although Kimmo systems can handle both prefixes and suffixes, most available Kimmo rule sets deal only with suffixes. The suffix-only case is dramatically simpler, because there is no ambiguity due to alternately adding prefixes and suffixes, and thus the task of parsing the result of the Kimmo analysis is trivial. A limitation of the Kimmo technique is that it requires the root words to be known in advance. It cannot analyze an apparently inflected form of an unknown word.

In contrast, the morphological mechanisms used in the Pilot indexer and the other indexing systems presented here handle not only prefixes and suffixes, but also words formed by lexical compounding of other known words (e.g., “bitmap”). They also handle both known and unknown roots, and include special patterns for forms such as e-mail addresses, date strings, Roman numerals, etc. Moreover, these rules, unlike those of Kimmo systems, allow the inclusion of conditions that test more than just the syntactic category of the root, even to the extent of testing semantic subsumption facts in the conceptual taxonomy. Some of the rules also make semantic subsumption assertions about analyzed words. Furthermore, unlike Kimmo systems, the morphological component described here imposes a preference ordering on the rules and selects a small set of preferred interpretations, with relative preferences among them, rather than generating all possible analyses.

6.8 Alternative approaches

Several of the problem areas addressed by the conceptual indexing systems presented here could be addressed in different ways. Some of these have already been covered in the introduction. In this section, I will discuss alternative approaches to the problem of morphological variation in queries and the problem of determining semantic relationships between terms of a query and those of a text.

6.8.1 Morphological variation

The classic technology in document retrieval systems for dealing with morphological variation is to use a “stemming” algorithm to truncate inflectional and derivational endings to produce “stems” such as “comput” for the various forms of “computer,” “computation,” and “computing.” However, these stemming algorithms can generate considerable “noise” in the retrieval process by collapsing and indexing together stems that come from unrelated words. For example, for one common stemming algorithm, “copper,” “cop,” “cope,” and “copulate” all stem to “cop.” Occasionally errors are made in the other direction as well when the rules for suffix removal don't end up producing the same stem for words that are in fact related. Moreover, stemming rules are traditionally applied to the indexed text and only the stems are indexed, thus losing information about the actual words in the material. This eliminates the ability for a user to later specify whether stemming is desired or not. In order to get any hits at all, stemming has to be consistently applied to both the indexed material and the query. Krovetz [1993] presents a good discussion of stemming algorithms

and the role of morphology in information retrieval. He presents experimental evidence for the benefits of doing full-fledged morphology and discusses an interaction between morphology and word-sense disambiguation.

Another alternative to morphology is the use of wildcards to match arbitrary suffixes of words. This method, however, can provide only a crude approximation to true morphological analysis. In particular, it fails to deal with irregular forms, it often results in undesired matches of unrelated terms, and it requires mental effort from the information seeker to decide whether and where to use the wildcard character.

6.8.2 Semantic distance

An alternative approach to conceptual indexing for making connections between concepts in a query and those in a text (besides synonym thesauri and hierarchical thesauri discussed earlier) is to use some form of "semantic distance" measure as a similarity metric to assign a score to a document with respect to a query. The elements of conceptual indexing we have presented do not include a measure of semantic distance that would assign a numerical measure of the semantic distance between concepts in the taxonomy and concepts in the text. Setting a threshold for semantic distance could allow queries to be effectively expanded to include all terms within a specified semantic distance from them. There are a number of ways that a semantic distance measure could be defined and used, and we have explored some of them with experiments. However, no single definition of semantic distance has seemed to have any particular claim to being more correct than another. Indeed, it appears that whether two concepts are similar or not depends on the purposes for which they are being compared. For example, a hammer is close to a hatchet for the purposes of pounding a nail, but not for the purposes of chopping down a tree. What might be a good measure of semantic similarity for one user's purposes might be ill-suited to the purposes of another.

In contrast, the conceptual indexing system imposes a *topology* on the space of concepts (i.e., concepts have neighborhoods), but there is no measure of distance between them. This approach gives the user control over the level of generality of a request in a way that is conceptually meaningful, unlike picking a numerical threshold value on a semantic distance scale that has no objective calibration. The taxonomic structure of a conceptual taxonomy provides not only a neighborhood for any concept, but (unlike a distance function) also provides an orientation in conceptual space. That is, the generality relation enables a searcher to distinguish up from down, so that the user can navigate in conceptual space using generality as an orientation. The conceptual taxonomy approach has a further advantage that the "neighborhoods" are conceptually characterized in a way that can be read and understood—not simply grouped into unlabeled clusters by imposing cutoffs on a distance metric.

6.9 Future directions

We have already discussed a number of ways that this technology can be applied to solve a variety of problems involving indexing topics in a reference work or catalog or in a collection of documents. These techniques are also applicable to indexing conceptual descriptions in many other applications. For example, conceptual subsumption techniques can be applied to organizing software methods for automatic selection and compilation in a variation of object oriented programming methodology called "taxonomic programming" (see,

e.g., [Woods, 1986]).

Conceptual taxonomies can also be used to organize large rule sets in AI *expert systems*, i.e., knowledge-based reasoning systems in which collections of rules are used to enable computers to carry out tasks that ordinarily require human expertise. For example, DEC's Xcon system automatically checks new system configuration specifications for consistency (and suggests additional required components as necessary) in order to eliminate unnecessary delays and expense when filling customer orders. (IBM also has such a system.) Another example is AT&T's ACE system, a failure diagnosis system that analyzes service interruption reports to identify possible sources of the problem and automatically generates work orders to dispatch service personnel to address the problem. Woods [1991] has more to say about applying conceptual taxonomies to such knowledge-based reasoning problems.

In the next few subsections, I will describe some additional areas of application of conceptual indexing technology.

6.9.1 Publishing conceptually indexed material

If you are an information publisher, then it is only a short step from being able to conceptually index and organize a body of material to deciding to publish that material with a pre-constructed index. This can make publishing online material much more useful to its intended audience in just the way that Sun's online documentation system, AnswerBook, publishes Sun's system documentation on a CD-ROM, packaged with a state-of-the-art search and retrieval system and a precomputed index.

When material is to be published with a precomputed conceptual index, there are some slightly different aspects to the indexing process (compared to indexing arbitrary collections of text) that can be exploited to improve the quality of the resulting index. For one thing, the domain and vocabulary of the material is known and one can make sure that the semantic subsumption axioms in the knowledge base are appropriate to that domain and vocabulary. Also, since one has control of the material, it is possible to consider annotating key ambiguities in the source material to indicate the intended interpretation (or one could actually make changes in the source material to avoid the ambiguity).

The ambiguity problem can also be reduced by choosing to ignore word senses and related subsumption axioms in the lexicon that do not occur in the material being indexed. Finally, rather than simply accepting phrase ambiguities in the index, it is possible to postedit the computed index or to annotate the source material to choose the correct interpretations and eliminate incorrect ones. The conceptual index can even support this process by identifying those phrases for which it has cross references to alternative interpretations. In addition, one can impose editorial selectivity on the resulting conceptual taxonomy to eliminate concept phrases that are not likely to be of any interest as topic descriptions (e.g., "next paragraph").

We have already done one small experiment in this direction in support of a project of the Speech Group at SunLabs in Chelmsford investigating the potential for an automated telephone-based catalog sales agent using computerized speech recognition and synthesis for the input and output. Conceptual indexing was used in this system to match concepts in a spoken request like "red sweater" with catalog items in the data base like "crimson drifter."

The general purpose semantic axioms in the Solar system were able to capture many of the desired inferences, but we quickly encountered the need to create application-specific restrictions on the use of these axioms to avoid unwanted interpretations such as "red" in the sense of "communist." Some additional infrastructure was added to our experimental system to support custom axioms and lexical entries to support this kind of application.

6.9.2 Taxonomic documentation

Software documentation tends to be organized around the structural elements of the software artifact—not what can be done with those artifacts. This is not the best organization for finding out how to do something. For example, in Sun's AnswerBook, the information about how to get the File Manager to display multiple windows occurs in a section titled "Moving a Folder onto the Workspace" (a description of the gesture rather than its purpose). The information about what this gesture is good for comes in the last sentence of the section, where it says that you can use this to display the contents of more than one folder at a time. Similarly, UNIX man pages are organized by the names of the commands, not by what you can do with them. Organizing this material by conceptual descriptions of actions to be achieved and using taxonomic subsumption technology to match statements of need with the described actions can be dramatically more effective.

Organizing documentation by descriptions of actions would also make it possible to index activities that otherwise don't make it into the documentation because there's no single command to do them. For example (with my apologies if you don't speak UNIX), a standard way to check how much memory is on a UNIX workstation is by "piping" the output of a command called `dmesg` through another command called `grep`—specifically, by typing:

```
/etc/dmesg | grep "mem"
```

There is no obvious place to put this kind of information in the man pages scheme of organization, except perhaps as a minor footnote in the page for `dmesg` or `grep`. One might include such information in a separate FAQ file, but the question then arises how to organize the FAQ file if the number of items in it is large. In a conceptual index, this "incantation" could be recorded under the description "see how much memory is on a workstation," which could be automatically classified under "see something" and automatically related to "amount of memory," "memory on a workstation," etc. Specific information on how to do things could then be looked up in the resulting *taxonomic documentation* by either navigating the conceptual taxonomy or by expressing a request to the dynamic passage retrieval algorithm.

6.9.3 Conceptually mediated hypertext

Using a conceptual index to cross-reference hypertext and hypermedia documents (see [Woods, 1995b]) can make it possible for collections of different documents, created with no knowledge of each other, to effectively have hypertext connections among them. By simply indexing them all into a single conceptual index, any phrase concept in any document can then serve as a hypertext link into the conceptual taxonomy, which in turn can link to related concepts and to occurrences of those concepts in other documents.

Importantly, such *conceptually mediated hypertext* allows a user to find out qualitative information about the concept corresponding to a link anchor by examining what's known about it in the conceptual taxonomy before deciding whether to actually link to the destination anchor. It also allows a convenient way to implement the equivalent of multiple link types by following explicit linking relationships between concepts to get to different labeled destinations. For example, a link from the first use of a term in a document would lead to the concept for that term in the taxonomy, from which various conceptually identified links could point explicitly to (1) a definition of the term, (2) places where the use of the term is described or the term is explained, (3) uses of the term in other documents, and (4) other concepts related to the term in the taxonomy. This process of conceptual mediation imposes some structure and semantics on the hypertext linking mechanism, which otherwise imposes no meaningful relationship between the description of a link anchor and a link destination.

6.9.4 Conceptually assisted Web browsing

Another application of conceptual indexing is an adjunct to browsing the World Wide Web using a Web browser and a search engine. The conceptual indexer is fast enough to dynamically index pages as you browse the Web, providing a conceptual index to everything you have encountered and making it easy to return to anything you have seen. This is a significant improvement over bookmarks as a way of finding your way back to places you remember. A Java-based application called BrowseGuide [Ambroziak, 1997] connects to any Java-enabled Web browser to dynamically index not only all of the pages that you visit, but also all of the pages that are one link removed from those pages, giving you a kind of "peripheral vision" when browsing the Web that attracts your attention to specific pages and specific passages within pages when anything matching your expressed interests is encountered. This is particularly useful when the page you are browsing is the hit list from a Web search request. The BrowseGuide then automatically indexes each of the pages found by the search engine and can take you directly to the pages and passages of most interest.

6.9.5 Other applications

The technique of conceptual indexing is well-suited to indexing descriptions of reusable code modules or library class definitions to help programmers find code modules that they can incorporate into their programs. This is useful when programming in languages like Java that have large class libraries and large communities of developers who share code modules.

In a similar way, conceptual indexing can be used to organize descriptions of complex, multifeatured products such as product offerings by financial institutions. Such indexes can play a key role in creating automatic network-based agents for matching customer needs with products offered. This can provide a key element of infrastructure for Web commerce.

7 Conclusions

This paper presents a methodology for indexing collections of material to help people find information more effectively. This is done by organizing conceptual descriptions that are automatically extracted from text into a *conceptual taxonomy* that can be browsed and searched. The index can be used by search and retrieval algorithms to make connections between concepts used in an information request and related concepts used in relevant passages of material. Experiments with this technology show that it can significantly improve people's ability to find relevant documents and to locate specific passages of text in response to specific information needs.

Using the structure of conceptual descriptions, together with basic subsumption facts about the generality of concepts, a "structural subsumption" algorithm can determine when one concept is more general than another. This structural subsumption relationship can be used to organize a collection of topic phrases into a conceptual taxonomy in which each concept is explicitly linked to its *most specific subsumers* (i.e., the most specific concepts in the taxonomy that are more general than the one in question). This taxonomy can then be used to support browsing and navigation in "conceptual space," and to make connections between terms used in a request and related terms that may be used in relevant passages of text.

The conceptual taxonomy escapes the limitations of traditional library classification hierarchies such as LC and Dewey by allowing conceptual categories to have multiple parent categories (i.e., their most specific subsumers), and it allows the automatic location of the most specific concepts that subsume an input request (i.e., are more general than the input request). This structure can be used by interactive browsers that focus on the conceptual "neighborhood" of an input request, even when the request does not occur literally in the taxonomy. This makes it possible to find out something about a topic without having to guess exactly how that topic is phrased in the taxonomy. The conceptual taxonomy also provides an orientation to conceptual space that allows an information seeker to control the level of generality of a request and to navigate effectively through the taxonomy to find related information.

An important feature of this technology is its ability to automatically determine where in a conceptual taxonomy a query phrase would belong when it does not already occur in the taxonomy. It does this by locating the most specific subsumers of the query phrase and locating its most general subsumees. Thus, when a request is not an exact match to a concept in a conceptual taxonomy, subsumption algorithms can determine if there are more general or more specific descriptions that relate to the request.

This conceptual taxonomy provides an oriented topology for the space of conceptual descriptions that not only can be browsed and navigated effectively by an information seeker, but also supports a *dynamic passage retrieval* algorithm that can locate specific passages of information in response to a request, taking into account variations in wording that may occur. It does this by using information from the conceptual taxonomy to make connections between terms in the request and more specific terms and other related terms that may occur in passages of interest. The dynamic passage retrieval algorithm identifies specific passages in the indexed material where the information requested may be found, and uses a *relaxation ranking* algorithm to assign penalty scores to these passages. These scores reflect the degree to which a passage deviates from an exact match to the input request.

Many observations about the value and benefits of elements of this technology have been made throughout this presentation, as the various features of this technology and the experiments we have done have been described. Since space precludes repeating them all here, I can only refer to their existence, hereby directing you to the previous presentation. If you want to more fully understand this technology, especially the way in which the various elements interact, you will just have to read the whole thing. }:-)

The methodology presented here has been embodied in a series of systems—Pilot, Recall, Solar, and Nova—which have been used to explore the merits of this technology. Experiments with these systems have shown that the structure of automatically-derived conceptual taxonomies can provide an information seeker with an intuitive and useful map of the way concepts and terminology are used in a body of material. Experiments also show that the dynamic passage retrieval algorithm is highly effective at locating specific passages in the material in response to descriptions of specific information needs, and that the relaxation ranking method used by the dynamic passage retrieval algorithm is especially effective at exploiting morphological and semantic subsumption information to improve information retrieval effectiveness.

Using the Recall system, which was designed to explore the capabilities of the dynamic passage retrieval algorithm, we indexed the UNIX man pages and measured the retrieval performance of this algorithm against a variety of traditional information retrieval techniques on a query set of spontaneously-worded natural English descriptions of information need. The query set consisted of mostly short specific requests and consequently posed a difficult challenge for traditional information retrieval techniques. The results of this experiment showed that the dynamic passage retrieval algorithm was substantially superior to all of the traditional methods tested, and the improvement was statistically significant at the .03% confidence level (i.e., the probability that the improvement was accidental was less than .0003).

Using the Solar system, a network server implementation of this technology with a Netscape/Mosaic interface, we have constructed taxonomies of considerable size (the largest being over a million concepts) and made several of them available to users to assess the utility of the technology. We have now applied these techniques to retrieval of online information from a variety of sources and exposed the technology to a variety of different kinds of users who have found the basic concepts of conceptual indexing and dynamic passage retrieval to be practical and useful. The Nova release, a C++ implementation, has been developed to be incorporated into applications.

In summary, we have presented a new approach to organizing and accessing online information. This approach, which combines techniques from natural language processing, knowledge representation, and traditional information retrieval, has been implemented, tested, and found useful. Unlike many attempts to use natural language processing to improve information retrieval, this approach has demonstrated significant benefit from using knowledge about the forms and meanings of words and semantic relationships among concepts to find connections between terms in a request and related concepts that may occur in relevant material. Conceptual indexing technology has many applications in a wide variety of information access applications. It has been shown that this technology can substantially improve people's ability to find specific answers to specific information needs. A summary of the principal benefits of this technology is given in Figure 11.

- Dynamic passage retrieval—finds specific passages of information content in response to specific requests
- Less time spent reading—produces a well-ranked graded list of specific passages within documents so you don't have to search through documents for the information you need
- Less time spent searching—automatic relaxation of query specifications and appropriate ranking of resulting passages reduces the number of queries required to find what you need
- Paraphrase tolerant—makes significant, effective use of linguistic and semantic knowledge to make connections from query terms to related wordings in text
- Conceptually maps the indexed material—automatically constructs a conceptual taxonomy that shows how terminology is used in the collection and how various concepts are related
- Provides conceptual feedback—a quick overview of related terminology in the conceptual taxonomy provides suggestions for improved query formulation
- Conceptual navigation—the conceptual taxonomy is suitable for effective browsing and navigation in conceptual space
- Value-adding technology—complements traditional information retrieval techniques
- Precision tool—works better than traditional techniques for short queries and small targets
- Effective use of knowledge—works well with a little knowledge and works better with more knowledge

Figure 11. Benefits of conceptual indexing technology.

8 References

- Allen, James [1987], *Natural Language Understanding*, Benjamin/Cummings, Menlo Park, Calif., 1987.
- Ambroziak, Jacek R. [1997], "Conceptually Assisted Web Browsing," Poster Presentation, 6th International World Wide Web Conference, April 1997, Santa Clara, Calif. (see: <http://poster.www6conf.org/poster/702/guide2.html>).
- Bookman, Lawrence A. and William A. Woods, "Adding Knowledge can Improve Information Search," *Sun Microsystems Laboratories Technical Report*. [forthcoming]
- Brachman, R. J. and J. Schmolze [1985], "An overview of the KL-One knowledge representation system," *Cognitive Science*, Vol. 9, 1985, pp. 171-216.
- Brill, Eric [1994], "Some Advances in Transformation-Based Part of Speech Tagging," *Proceedings of Twelfth National Conference on Artificial Intelligence (AAAI-94)*, AAAI Press, Menlo Park, Calif., 1994, pp. 722-727.
- Fallows, James [1996], "Navigating the Galaxies," *The Atlantic Monthly*, April 1996, pp. 104-107. (for an online version, see: <http://www.theatlantic.com/atlantic/issues/96apr/computer/computer.htm>)
- Koskenniemi, Kimmo [1983], "Two-level morphology for morphological analysis," *Proceedings of The Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, Morgan Kaufmann Publishers, Los Altos, Calif., 1983, pp. 683-685.
- Krovetz, Robert [1993], "Viewing Morphology as an Inference Process," *Proceedings of the Sixteenth International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, PA, June 27-July 1, 1993 (SIGIR '93), ACM Press, 1993, pp. 191-202.
- Lenat, D. B. and R. V. Guha [1990], *Building Large Knowledge-Based Systems: Representation and Inference*, Addison-Wesley, Reading, Mass., 1990.
- Miller, G. A. [1995], "WordNet: A Lexical database for English," *Comm. ACM*, November, 1995, pp. 39-41.
- Northcutt, Duane [1995], "Interactive Services Technologies," in Sun Microsystems Laboratories Staff (eds.), *Fiscal 1995 Project Portfolio Report*, TR-95-50, Sun Microsystems Laboratories, Mountain View, Calif., November, 1995. (for an online version, see: <http://www.sunlabs.com/techrep/1995/annualreport95>)
- Salton, Gerard [1989], *Automatic Text Processing*, Addison-Wesley, Reading, Mass., 1989.

Salton, Gerard, J. Allan, and C. Buckley [1993], "Approaches to Passage Retrieval in Full Text Information Systems," *Proceedings of the Sixteenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR '93)*, ACM Press, 1993, pp. 49-58.

Sowa, John F. [1983], *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, Mass., 1983.

Sproat, R. [1992], *Morphology and Computation*, MIT Press, Cambridge, Mass., 1992.

Woods, W. A. [1970], "Transition Network Grammars for Natural Language Analysis," *Comm. ACM*, 13, No. 10, October, 1970. Reprinted in Yoh-Han Pao and George W. Ernest (eds.), *Tutorial: Context-Directed Pattern Recognition and Machine Intelligence Techniques for Information Processing*. IEEE Computer Society Press, Silver Spring, MD, 1982, and in Barbara Grosz, Karen Sparck Jones and Bonnie Webber (eds.), *Readings in Natural Language Processing*, Morgan Kaufmann, San Mateo, Calif., 1986.

Woods, W. A. [1986], "Important Issues in Knowledge Representation," *Proceedings of the IEEE*, Vol. 74, No. 10 (October, 1986). Reprinted in Peter G. Raeth (ed.), *Expert Systems: A Software Methodology for Modern Applications*, IEEE Computer Society Press, Los Alamitos, Calif., 1990.

Woods, W. A. [1987], "Grammar, Augmented Transition Network," in Stuart C. Shapiro (ed.), *Encyclopedia of Artificial Intelligence*, Wiley, 1987, pp. 323-333.

Woods, W. A. [1991] "Understanding Subsumption and Taxonomy: A Framework for Progress," in John Sowa (ed.), *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann, San Mateo, Calif., 1991, pp. 45-94.

Woods, W. A. and James G. Schmolze [1992] "The KL-ONE Family," *Computers & Mathematics with Applications*, Vol. 23, No.'s 2-5, (January-March, 1992), Special Issue on Semantic Networks in Artificial Intelligence, Part 1, pp. 133-177. Also reprinted in Fritz Lehmann (ed.), *Semantic Networks in Artificial Intelligence*, Pergamon Press, 1992.

Woods, W. A. [1995a], "Guidelines for Precision Content Retrieval," *Sun Microsystems Laboratories Technical Report*. [forthcoming]

Woods, W. A. [1995b] "Content-Based Information Access using Structured Conceptual Taxonomies," slides presented at the Fourth International World Wide Web Conference, Boston, MA, December, 1995. (for a summary, see: <http://www.ai.mit.edu/projects/iiip/conferences/www95/woods.html>)

9 Glossary

AI—an acronym standing for "artificial intelligence" referring to methodologies for programming computers to perform mechanical reasoning activities such as diagnosis, planning, problem solving, and natural language understanding.

AnswerBook(TM)—an online documentation system developed by Sun Microsystems that makes system documentation available to users in a familiar hierarchical document structure and also accessible via a state-of-the-art search and retrieval system. This is the vehicle by which Sun distributes its system documentation on a CD-ROM instead of paper documents.

ATN grammar—a formalism for expressing grammars for natural (or artificial) languages (or a grammar in this formalism). The term "ATN" stands for "Augmented Transition Network," which is the device by which this formalism expresses grammars. An ATN resembles the transition diagram of a *finite-state recognizer* in that it consists of a network of alternative sequences of transitions from an initial state to some final state, each path through which corresponds to a possible "sentence" acceptable to the grammar. Unlike finite state recognizers, however, a transition in an ATN can represent the analysis of an entire subphrase of the input sequence, analyzed by a potentially recursive invocation of a subnetwork of the ATN. Additionally, the transitions in an ATN can be augmented with conditions and actions that specify computational constraints on when a transition can be taken and actions that will make incremental contributions to the structural analysis of the input. ATN grammars are an efficient grammar formalism that can capture the most subtle effects of natural language syntax and also integrate arbitrary formal sublanguages.

axiom—a basic known fact, specifically a subsumption fact, on which the testing of subsumption by the structural subsumption algorithm rests. We refer to these basic subsumption relationships (e.g., the fact that a car is a kind of automobile) as "subsumption axioms." These axioms are created by human lexicographers (or in some cases by morphological rules) and represented in a lexicon or an initially constructed taxonomy. In addition to a large collection of general purpose subsumption axioms that apply uniformly across many domains, a system can include various specialized axioms that are appropriate to specialized subject areas.

BBN—a research company in Cambridge, Massachusetts, one of the original developers of the ARPANET (now Internet). Full name: Bolt, Beranek and Newman Inc., with a division named BBN Systems and Technologies.

Boolean retrieval system—an information retrieval system in which query terms are combined with the Boolean operators AND, OR, and NOT, and documents are either retrieved or not depending on whether they satisfy the Boolean requirement. E.g., if the query is "book AND candle" then documents will be retrieved if and only if they contain both the words "book" and "candle."

canonical form—a standard form used to test items for equivalence by mapping each of them to their respective "canonical" forms, and then checking to see if the two canonical forms are the same. For example, comparing inflected words for equivalence by checking to see if their root forms are the same or comparing two lists of elements for set equivalence

by first sorting each of them to produce sorted canonical forms (which will be the same if and only if the two sets are equal).

cascaded indexing—a two-stage process consisting of a search in a traditional document retrieval system using high recall mode (i.e., using parameter settings chosen for high recall and therefore tolerating low precision), followed by a conceptual index of the results. This enables an information seeker to more effectively locate the information sought.

classification algorithm—an algorithm for determining where in a conceptual taxonomy a new concept belongs. Usually this is a combination of an MSS algorithm that determines the parent concepts under which the new concept should be inserted and an MGS algorithm that finds most general subsumee concepts that should become its children.

concept assimilator—a program module that adds words and phrases into a conceptual taxonomy and determines any other related concepts and conceptual relationships that should be added, such as more general concepts indicated by subsumption axioms or morphological relationships.

ConceptStore—a system for storing and using conceptual taxonomies developed at Sun Microsystems Laboratories as part of the conceptual indexing project. It constructs a persistent database for conceptual taxonomies, lexicons, and terminologies and provides facilities for classification of concepts, indexing concept positions in text, and retrieval of passages using the relaxation ranking algorithm. Used in conjunction with Textbase for the Nova conceptual indexing system.

conceptual indexing—a technique for extracting conceptual phrases from source material and assimilating those concepts into a conceptual taxonomy that can subsequently be used as a conceptual map of the material to support navigation of the material in "conceptual space" and to support access to the material by search and retrieval algorithms.

conceptual taxonomy—a hierarchical organization of conceptual descriptions organized by a relationship of generality. Each concept in a conceptual taxonomy has links to its *most specific subsumers* (its "parents" in the taxonomy) and links from its *most general subsumees* (its "children" in the taxonomy).

coreference—refers to the situation when two linguistic terms or descriptions must refer to the same thing. For example, in the sentence "John hurt himself," the two expressions "John" and "himself" corefer (i.e., must refer to the same thing). Similarly in the equation $3 + x = x + y$, the two occurrences of x corefer.

derivational operation—a *morphological operation* that derives a new (derived) word from one or more other words. In English, words are frequently derived from other words by adding prefixes or suffixes or by *lexical compounding*, an operation that combines two or more other words into a single word.

disambiguate—a technical term from computational linguistics that refers to the process of determining the correct interpretation of an ambiguous word, phrase or sentence, i.e., resolving the ambiguity.

Docformat—a text markup language used in some UNIX environments to prepare documentation.

document retrieval—the traditional form of information retrieval in which the result of a query is a list of documents considered relevant to the query. This is contrasted with *fact retrieval*, which directly answers questions, and *passage retrieval*, which identifies relevant passages within documents.

dynamic passage retrieval—a search and retrieval algorithm that can use a conceptual taxonomy to make connections between terminology used in a request and related terminology used in indexed material and can use positional location in the index to locate specific passages of text where answers to a request are likely to be found. Unlike other passage retrieval algorithms, *dynamic* passage retrieval uses location information to identify specific passages at query time, and the sizes of the resulting passages are variable, depending on how much must be included to identify a possibly relevant hit.

Emacs—a text editor program developed in the academic research community to edit a wide variety of different kinds of text files and program source code with a powerful set of operators associated with command keys.

expert system—a computer program that embodies some element of human knowledge or expertise in a form that can be automatically applied to problems.

extension—a term from philosophy that refers to the collection of entities denoted by a referring description. This is a technical sense of the ordinary English word.

extensional semantics—a term from philosophy that refers to a kind of semantics in which the meaning of a description is identified with the collection of things described. This collection is called the *extension* of the description.

fact retrieval—fact retrieval systems (also called question answering systems) deliver specific answers in response to requests that are fully analyzed both syntactically and semantically and processed against a fully-analyzed database.

finite-state recognizer—a term from automata theory that refers to a finite-state machine (actually an abstract automaton, not a physical machine) that recognizes sequences of elements of a class by following sequences of transitions corresponding to successive elements of the input (through a network of alternative possible sequences of such transitions) from a distinguished initial node to some one of a specified set of final nodes. An input sequence of elements is considered a "sentence" acceptable to the recognizer if there is such a path from the initial node to some final node. The term "finite-state" comes from the fact that the abstract computation performed for each successive element of the input (in order to determine what transitions are possible) depends only on the current input element and the current node in the finite-state network. Thus, the entire state of the computation at this point (i.e., the only memory of the analysis of the input sequence up to this point) is represented by the current node in the transition network. (Accordingly, these nodes are usually called "states," and the network of nodes is called a "state transition network.") Since there are only a finite number of nodes in the transition network, the computation at each point depends on only a finite amount of information.

grammar—a codification of rules for the syntactic structure of a language. As used here, it refers to a codification of such rules in a form that can be used by a *parser*.

hierarchical thesaurus—a hierarchical ordering of words or phrases used to expand queries in certain information retrieval systems. When a term in a query occurs in the thesaurus, all of the terms that occur below that term in the hierarchical ordering are added to the query as alternatives.

hit—a term commonly used to refer to an item retrieved by a retrieval algorithm—a document, for a document retrieval system, or a specific retrieved passage for a passage retrieval system. A hit may be augmented with additional information about the nature of the match.

inflection—the *morphological operation* that alters the form of a word to indicate *syntactic features* such as plurality, gender, number, tense and aspect (e.g., the operation that forms the plural form "dogs" from "dog" by adding the suffix "s").

information retrieval—a technology for finding information from indexed material in response to a query. Although usually identified *with document retrieval*, in its broad sense, information retrieval also includes *passage retrieval* and *fact retrieval* technology.

intensional structure—a technical term from philosophy that refers to a structure that represents the meaning of a sentence or phrase. Note that this is different from *intentional*—it's not a misspelling.

intensional subsumption—a criterion for subsumption based on the *intensional structure* of concepts rather than their extensions.

Internet—a global network of computers in many different organizations and many different countries, linked together by a packet-switched communications protocol. This network forms the infrastructure for the World Wide Web.

intranet—a computer network linking together various computers within an organization.

Kimmo system—a system for expressing morphological rules that uses finite-state transducers to represent the bidirectional transformations between roots of words and their derived and inflected forms. This method is also called "two-level morphology" (see [Koskenniemi, 1983]).

KL-One—a knowledge representation system that pioneered the use of automatic classification of concepts based on an understanding of the structure of their meaning (see [Woods and Schmolze, 1992]).

knowledge technology—technology for helping people deal with knowledge—acquiring it, analyzing it, organizing it, disseminating it, and using it.

knowledge technology group—a research group at Sun Microsystems Laboratories devoted to the discovery and development of useful knowledge technology. The conceptual indexing technology presented here was developed by this group.

lexical compounding—the *morphological operation* of forming a new word by concatenating two or more other words into a single word, as in "shoelace."

man pages —the system of online documentation used in UNIX environments to organize documentation of the extensible set of UNIX commands. Each command in UNIX has an associated man page that specifies what the command does and how to use it. Anyone who contributes a new command to UNIX's command set is expected to also produce a man page for that command that will be included in the man page collection. The name "man page" is a shorthand for "manual pages," i.e., pages of a manual. A UNIX command called "man" can be used to invoke a display of the man page for any specified command.

MGS—an acronym standing for "most general subsumee."

morphological operation—an operation that alters the form of a word to indicate inflectional information or that derives a new word from one or more existing words.

morphology—the system of formation of words in a language, including inflection, derivation, and compounding. In English, inflection is usually indicated by suffixes added to the end of the word, and new words can be derived by adding prefixes or suffixes to words or by the *lexical compounding* of two or more words into a single word.

Mosaic—a network Web browser that provides a convenient interface for accessing information over a computer network.

most general subsumee —a most general subsumee (MGS) of a concept *x* is a more specific concept that is not *subsumed* by any subsumee of *x*. The most general subsumees of a concept are its "children" in a conceptual taxonomy and are displayed below it. A concept can have many most general subsumees (children).

most specific subsumer—a most specific subsumer (MSS) of a concept *x* is a subsumer of *x* that does not subsume any other concept in the taxonomy that subsumes *x*. The most specific subsumers of a concept are its "parents" in a conceptual taxonomy and are displayed above it. A concept can have more than one most specific subsumer—i.e., concepts can have more than one parent in a conceptual taxonomy.

MSS—an acronym standing for "most specific subsumer."

narrowing—a term used by the information retrieval community to denote the process of refining a query to exclude unwanted hits when there are too many distractors in the results list (opposite of "broadening" a search).

Netscape—a commercial second generation version of *Mosaic*.

Nova—a conceptual indexing and retrieval system developed by Sun Microsystems Laboratories. Composed of *Textbase*, which performs the text analysis and word and phrase extraction, and *ConceptStore*, which stores the conceptual index and lexicons and provides passage retrieval capability.

parser—a computer program that can analyze a sentence or phrase and construct a representation of its syntactic structure that can be used in later stages of processing to find major constituents and relationships among them.

part-of-speech tagging—a process of automatically assigning part-of-speech categories (syntactic word classes) to words in text based on their context of occurrence with other words, usually by methods that do not involve full parsing of the sentences—an attempt to obtain some useful syntactic information with processes that are more efficient than full parsing.

passage retrieval—an information retrieval term that refers to retrieving passages of material within a document rather than documents as a whole, or to retrieving documents based on finding good passages within them rather than aggregating information from the document as a whole.

precision—one of the two traditional measures of information retrieval performance. Precision measures the percentage of retrieved documents that are judged to be relevant to the original request—i.e., the ratio of the number of relevant retrieved documents over the total number of retrieved documents.

Precision Content Retrieval—the combination of conceptual indexing technology with dynamic passage retrieval using the relaxation ranking retrieval algorithm.

quantificational tags—tags associated with roles of a conceptual description (or slots in a frame or fields in a data record) that explicitly indicate the quantificational import of that role—e.g., whether the specified value names the exact value of that role, or specifies a constraint that the value of that role must satisfy, or a constraint that some value of that role must satisfy, etc.

question answering system—a computer system that delivers specific answers to direct questions (also called a *fact retrieval system*).

recall—one of the two traditional measures of information retrieval performance. Recall measures the percentage of relevant documents in a collection that are actually found by a retrieval system—i.e., the ratio of the number of relevant retrieved documents over the total number of relevant documents contained in the collection.

Recall—one of the experimental conceptual indexing and retrieval systems developed by the knowledge technology group at Sun Microsystems Laboratories. The Recall system was developed to explore the capabilities of conceptual indexing and retrieval technology compared to those of traditional document retrieval systems.

relaxation ranking—a method of rating retrieved passages in a passage retrieval algorithm by penalty scores that measure how much the conditions of an exact match with the query (i.e., the exact terms, in the same inflected form, in the same order and the same syntactic relationships, and with no intervening words) must be "relaxed" in order to accept the match.

relevance feedback—a technique whereby a user tells the retrieval system which documents from an initial retrieved set are good and which ones are not, so that the system can take that information into account and try again. In the second round of a relevance feedback search, the system attempts to find documents that are more like the good ones and less like the bad ones. This technique is one of the major discoveries of information retrieval research, but is not available in many commercial systems.

roles —the relationships that are filled by constituents of a structured concept. Roles in a concept are analogous to fields in a data record (in database terminology) or slots in a frame (in AI terminology), except that they reflect more the structure of natural concepts rather than artificial data structures, and they may contain attributes such as quantificational tags that are not normally associated with such data structures (but should be for expressive adequacy).

Scribe—a text markup language used in some UNIX environments to prepare documentation and other publications.

SearchIt—a commercial indexing and retrieval product developed at Sun Microsystems using a search engine from Fulcrum Technologies, a major supplier of commercial search and retrieval technology.

semantics—a term from philosophy referring to the meaning of a sentence or phrase. The term is also used to denote the science of the study of meaning, but that is not the sense used in this paper.

Solaris—the standardized UNIX operating system developed by Sun Microsystems to provide a single consistent operating system interface that is the same on both Sun and Intel processors.

stemming—a technique used in some information retrieval systems to replace different forms of a word with a canonical form of that word called its "stem." This stem is determined by an algorithm that repeatedly removes recognizable suffixes from the end of a word until no more can be removed. For example, "compute," "computer," and "computation" would all have the same stem, "comput." The theory is that when stems are indexed instead of the original words, and terms in a query are stemmed before lookup, then retrieval will be independent of the form of the word. Unfortunately, stemming algorithms sometimes reduce fundamentally-different words to the same stem (e.g., copper, cop, cope, and copulate) and occasionally reduce words to different stems when they should be the same.

structural subsumption—a criterion for determining whether one concept *subsumes* another (i.e., is more general) by virtue of the structures of the concepts and the subsumption relationships among their constituent elements.

subsumption—the relationship of generality between concepts in which a term X *subsumes* a term Y if X is more general than Y, or equivalently, if Y is more specific than X. The more general of a pair of such terms is the *subsumer* and the more specific is the *subsumee*. A subsumer *generalizes* a subsumee, and a subsumee *specializes* a subsumer. A *most specific subsumer* (MSS) of a concept X is a subsumer of X that does not subsume any other

subsumer of X. Similarly, a *most general subsumee* (MGS) is a subsumee that is not subsumed by any other subsumee.

subsumption axiom—see axiom.

SunExpress—a division of Sun Microsystems, Inc. that sells software and hardware by catalog and over the telephone.

SunSolve Online—a product of Sun Microsystems SunService division that delivers an extensive collection of reference material useful to UNIX system administrators with a full text search capability. This material is also available in the form of a CD-ROM, SunSolve CD-ROM.

synonym thesaurus—a collection of sets of words used to expand queries in certain information retrieval systems. When a term in a query occurs in a synonym set in the thesaurus, all of the other terms of that synonym set are added to the query as alternatives.

syntactic features—features of words and phrases that govern how they can be used in grammatical sentences (also called grammatical features) and/or features of structural elements of sentences that characterize how they were analyzed and how they can participate in larger elements. Examples of syntactic features of words include plurality, gender, number, tense and aspect, and codes indicating whether verbs take direct objects, etc. Examples of syntactic features of elements of a sentence include indicators of what the subject of a sentence is, what the determiner of a noun phrase is, whether a sentence is active or passive, whether it is a question or a statement, etc.

taxonomy—a classification of categories of things.

terminological logic—a name given to the research area dealing with the logic of subsumption of conceptual descriptions in knowledge representation systems.

Textbase—a collection of text processing components developed at Sun Microsystems Laboratories that deal with processing texts, performing lexical analyses on their words and extracting words and phrases to be indexed. Used in conjunction with *ConceptStore* for the *Nova* conceptual indexing system.

tf.idf—the term "tf.idf" refers to a standard information retrieval method that weights term occurrences in a document by the number of times they occur in the document and normalizes them by the number of documents that they occur in. The name denotes the product of term frequency (tf), which measures the number of occurrences of a term in a document, times the inverse document frequency (idf), which is 1 divided by the number of documents in the indexed material in which the term in question occurs. This basic equation characterizes a family of traditional retrieval methods that differ in detail, but share this basic strategy.

thesaurus—a mechanism used by many information retrieval systems to provide automatic term expansion of queries. A thesaurus can be either of two principal kinds—a *synonym thesaurus* or a *hierarchical thesaurus*.

tokenizer—a computer program that segments a stream of characters into tokens that can be processed as units.

topology—a mathematical system in which elements have neighborhoods (i.e., groups of other elements that are "near" them), but may not have distances or directions to other elements.

Troff—a text markup language used in some UNIX environments to prepare documentation and other publications.

UNIX—UNIX is an open-systems standard computer operating system that is noted for its powerful capabilities and an extensible command-line interface that includes an ability to direct the output of one command into the input of another by a mechanism called "pipes." UNIX *man pages* are a standard way to organize online reference documentation for the evolving set of UNIX commands. Most of the network servers on the Internet run UNIX, and versions of UNIX, such as Sun's Solaris, are available on most hardware platforms.

vector-based retrieval—information retrieval based on the model of a text item as a vector with a position for each word of the language and an entry in that position representing the number of times the word occurs in the text item. This is sometimes referred to as a "bag of words" model, since it gives up all information about word order and proximity. A bag is a mathematical object that is like a set except that it also keeps track of how many times each element occurs in the set, and otherwise does not keep track of the order of the elements or any relationships between them (analogous to ordinary bags).

Venn diagram—Venn diagrams are diagrams consisting of overlapping circles used to illustrate the relationship between the Boolean operators AND and OR to set intersection and set union.

webcrawler—a program that traverses a portion of a Web computer network and gathers information such as an index of what words occur in what pages.

wildcard—wildcards are special characters that can be used in a query term to allow a match to any character or any substring of characters. Typically "?" is used to allow a match to any single character and "*" is used to allow a match to any number of characters.

10 About the Author

William A. Woods is a Principal Scientist and Distinguished Engineer at Sun Microsystems Laboratories in Chelmsford, Massachusetts. He is internationally known for his research in natural language processing, continuous speech understanding, and knowledge representation and is currently interested in technology for improving people's access to on-line information. He earned his doctorate at Harvard University, where he served as an Assistant Professor and later as a Gordon McKay Professor of the Practice of Computer Science. He is a past president of the Association for Computational Linguistics, a Fellow of the American Association for Artificial Intelligence, and a Fellow of the American Association for the Advancement of Science.

Dr. Woods worked at Bolt Beranek and Newman Inc. (BBN) when the Internet (then Arpanet) was being invented, and he built one of the first natural language question answering systems to answer questions about the Apollo 11 moon rocks for the NASA Manned Spacecraft Center. He was Principal Investigator for BBN's work in natural language processing and knowledge representation and for its first project in continuous speech understanding. Subsequently, he was Principal Scientist for Applied Expert Systems, Inc. and Principal Technologist for On Technology Inc.