The Sequoia 2000 Electronic Repository

Ray R. Larson
Christian Plaunt
Allison G. Woodruff
Marti A. Hearst

A major effort in the Sequoia 2000 project was to build a very large database of earth science information. Without providing the means for scientists to efficiently and effectively locate required information and to browse its contents, however, this vast database would rapidly become unmanageable and eventually unusable. The Sequoia 2000 Electronic Repository addresses these problems through indexing and retrieval software that is incorporated into the POSTGRES database management system. The Electronic Repository effort involved the design of probabilistic indexing and retrieval for text documents in POSTGRES, and the development of algorithms for automatic georeferencing of text documents and segmentation of full texts into topically coherent segments for improved retrieval. Various graphical interfaces support these retrieval features.

Global change researchers, who study phenomena that include the Greenhouse Effect, ozone depletion, global climate modeling, and ocean dynamics, have found serious problems in attempting to use current information systems to manage and manipulate the diverse information sources crucial to their research.[1] These information sources include remote sensing data and images from satellites and aircraft, databases of measurements (e.g., temperature, wind speed, salinity, and snow depth) from specific geographic locations, complex vector information such as topographic maps, and large amounts of text from a variety of sources. These textual documents range from environmental impact reports on various regions to journal articles and technical reports documenting research results.

The Sequoia 2000 project brought together computer and information scientists from the University of California (UC), Digital Equipment Corporation, and the San Diego Supercomputer Center (SDSC), and global change researchers from UC campuses to develop practical solutions to some of these problems.[2] One goal of this collaboration was the development of a large-scale (i.e., multiterabyte) storage system that would be available to the researchers over high-speed network links. In addition to storing massive amounts of data in this system, global change researchers needed to be able to share its contents, to search for specific known items in it, and to retrieve relevant unknown items based on various criteria. This sharing, searching, and retrieving had to be done efficiently and effectively, even when the scale of the database reached the multiterabyte range.

The goal of the Electronic Repository portion of the Sequoia 2000 project was to design and evaluate methods to meet these needs for sharing, searching, and retrieving database objects (primarily text documents). The Sequoia 2000 Electronic Repository is the precursor of several ongoing projects at the University of California, Berkeley, that address the development of digital libraries.

For repository objects to be effectively shared and retrieved, they must be indexed by content. User interfaces must allow researchers to both search for items based on specific characteristics and browse the repository for desired information. This paper summarizes the research conducted in these areas by the Sequoia 2000 project participants. In particular, the paper describes the Lassen text indexing and retrieval methods developed for the POSTGRES database system, the GIPSY system for automatic indexing of texts using geographic coordinates based on locations mentioned in the text, and the TextTiling method for improving access to full-text documents.

Indexing and Retrieval in the Electronic Repository

The primary engine for information storage and retrieval in the Sequoia 2000 Electronic Repository is the POSTGRES next-generation database management system (DBMS).[3] POSTGRES is the core of the DBMS-centric Sequoia 2000 system design. All the data used in the project was stored in POSTGRES, including complex multidimensional arrays of data, spatial objects such as raster and vector maps, satellite images, and sets of measurements, as well as all the full-text documents available. The POSTGRES DBMS supports user-defined abstract data types, user-defined functions, a rules system, and many features of object-oriented DBMSs, including inheritance and methods, through functions in both the query language, called POSTQUEL, and conventional programming languages. The POSTQUEL query language provides all the features found in relational query languages like SQL and also supports the nonrelational features of POSTGRES. These features give POSTGRES the ability to support advanced information retrieval methods.

We used these features of POSTGRES to develop prototype versions of advanced indexing and retrieval techniques for the Electronic Repository. We chose this approach rather than adopting a separate retrieval system for full-text indexing and retrieval for the following reasons:

1. Text elements are pervasive in the database, ranging in size from short descriptions or comments on other data items to the complete text of large documents, such as environmental impact reports.
2. Text elements are often associated with other data items (e.g., maps, remote sensing measurements, and aerial photographs), and the system must support complex queries involving multiple data types and functions on data.
3. Many text-only systems lack support for concurrent access, crash recovery, data integrity, and security of the database, which are features of the DBMS.
4. Unlike many text retrieval systems, DBMSs permit ad hoc querying of any element of the database, whether or not a predefined index exists for that element.

Moreover, there are a number of interesting research issues involved in the integration of methods of text retrieval derived from information retrieval research with the access methods and facilities of a DBMS. Information retrieval has dealt primarily with imprecise queries and results that require human interpretation to determine success or failure based on some specified notion of relevance. Database systems have dealt with precise queries and exact matching of the query specification. Proposals exist to add probabilistic weights to tuples in relations and to extend the relational model and query language to deal with the characteristics of text databases.[4,5] Our approach to designing this prototype was to use the features of the POSTGRES DBMS to add information retrieval methods to the existing functionality of the DBMS. This section describes the processes used in the prototype version of the Lassen indexing and retrieval system and also discusses some of the ongoing development work directed toward generalizing the inclusion of advanced information retrieval methods in the DBMS.[6]

Indexing

The Lassen indexing method operates as a daemon invoked whenever a new text item is appended to the database. Several POSTGRES database relations (i.e., classes, in POSTGRES terminology) provide support for the indexing and retrieval processes. Figure 1 shows these classes and their logical linkages. These classes are intended to be treated as system-level classes, which are usually not seen by users.

The wn_index class contains the complete WordNet dictionary and thesaurus.[7] It provides the normalizing basis for terms used in indexing text elements of the database. That is, all terms extracted from data elements in the database are converted to the word form used in this class. The POSTQUEL statement defining the class is

```
create wn_index (
termid = int4,      /* unique term ID */
word = text,        /* the term or phrase */
pos = char,         /* WordNet part of speech
                       information */
sense_cnt = int2,   /* number of senses of word */
ptruse_cnt = int2,  /* types and locations of */
offset_cnt = int2,  /* related terms in WordNet*/
ptruse = int2[] ,   /* database are stored in */
offset = int4[])    /* these arrays
```

All other references to terms in the indexing process are actually references to the unique term identifiers (*termid*) assigned to words in this class. The wn_index dictionary contains individual words and common phrases, although in the prototype implementation, only single words are used for indexing purposes. The other parts of the record include WordNet database information such as the part of speech (*pos*) and an array of pointers to the different senses of the word.

The kw_term_doc_rel class provides a linkage between a particular text item in any class or text large object (we will refer to either as documents) and

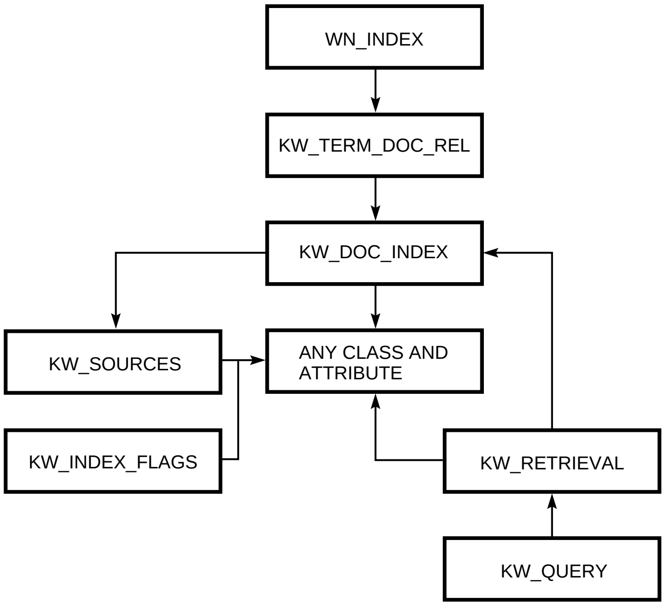


Figure 1
The Lassen POSTGRES Classes for Indexing and Their Linkages

a particular term from the wn_index class. The POSTQUEL definition of this class is

```
create kw_term_doc_rel (
termid = int4,    /* WordNet termid number */
synset = int4,    /* WordNet sense number */
docid = int4,     /* document ID */
termfreq = int4)  /* term frequency within
                     the document */
```

The raw frequency of occurrence of the term in the document (*termfreq*) is included in the kw_term_doc_rel tuple. This information is used in the retrieval process for calculating the probability of relevance for each document that contains the term. The kw_doc_index class stores information on individual documents in the database. This information includes a unique document identifier (*docid*), the location of the document (the class, the attribute, and the tuple in which it is contained), and whether it is a simple attribute or a large object (with effectively unlimited size). The kw_doc_index class also maintains additional statistical information, such as the number of unique terms found in the document. The POSTQUEL definition is as follows:

```
create kw_doc_index (
docid = int4,      /* document ID */
reloid = oid,      /* oid of relation
                      containing it */
attroid = oid,     /* attribute definition of
                      attr containing it */
attrnum = int2,    /* attribute number of attr
                      containing it */
tupleid = oid,     /* tuple oid of tuple
                      containing it */
sourcetype = int4, /* type of object -- attribute
                      or large object */
doc_len = int4,    /* document length in words */
doc_ulen = int4)   /* number of unique words in
                      document */
```

The kw_sources class contains information about the classes and attributes indexed at the class level, as well as statistics such as the number of items indexed from any given class. The following POSTQUEL statement defines this class:

```
create kw_sources (
relname = char16,    /* name of indexed
                        relation */
reloid = oid,        /* oid of indexed
                        relation */
attrname = char16,   /* name of indexed
                        attribute */
attroid = oid,       /* object ID of indexed
                        attribute */
attrnum = int2,      /* number of indexed
                        attribute */
attrtype = int4,     /* attribute type -- large
                        object or otherwise */
num_indexed = int4,  /* number of items
                        indexed */
last_tid = oid,      /* oid and time for last */
last_time = abstime, /* tuple added */
tot_terms = int4,    /* total terms from all
                        items */
tot_uterms = int4,   /* total unique terms from
                        all items */
include_pat = text,  /* simple patterns to */
exclude_pat = text)  /* match for indexable
                     /* items */
```

The other classes shown in Figure 1 relate to the indexing and retrieval processes. The Lassen prototype uses the POSTGRES rules system to perform such tasks as storing the elements of the bibliographic records in an appropriate normalized form and to trigger the indexing daemon.

Defining an attribute in the database as indexable for information retrieval purposes (i.e., by appending a new tuple to the kw_sources definition) creates a rule that appends the class name and attribute name to the kw_index_flags class whenever a new tuple is appended to the class. Another rule then starts the indexing process for the newly appended data. Figure 2 shows this trigger process.

The indexing process extracts each unique keyword from the indexed attributes of the database and stores it along with pointers to its source document and its frequency of occurrence in kw_term_doc_rel. This process is shown in Figure 3. The indexing daemon and the rules system maintain other global frequency information. For example, the overall frequency of occurrence of terms in the database and the total number of indexed items are maintained for retrieval processing. The indexing daemon attempts to perform any outstanding indexing tasks before it shuts down. It also updates the kw_doc_index tuple for a given indexable class and attribute with a time stamp for the last item indexed *(last_tid* and *last_time*). This permits ongoing incremental indexing without having to reindex older tuples.

Retrieval

The prototype version of Lassen provides ranked retrieval of the documents indexed by the indexing daemon using a probabilistic retrieval algorithm. This algorithm estimates the probability of relevance for each document based on statistical information on term usage in a user's natural language query and in the database. The algorithm used in the prototype is based on the staged logistic regression method.[8]

A POSTGRES user-defined function invokes ranked retrieval processing. That is, from a user's perspective, ranked retrieval is performed by a simple function call (kwsearch) in a POSTQUEL query language statement. Information from the classes created and maintained by the indexing daemon are used to estimate the probability of relevance for each indexed document. (Note that the full power of the POSTQUEL query language can also be used to perform conventional Boolean retrieval using the classes created by the indexing process and to combine the results of ranked retrieval with other search criteria.) Figure 4 shows the process involved in the probabilistic ranked retrieval from the repository database.

The actual query to the Lassen ranked retrieval process consists simply of a natural language statement of the searcher's interests. The query goes through the

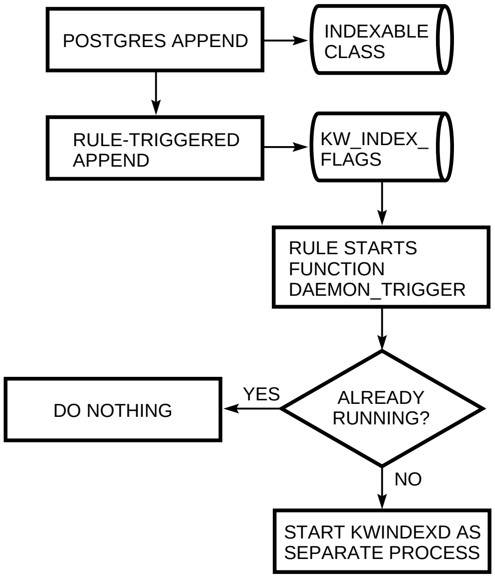


Figure 2
The Lassen Indexing Trigger Process

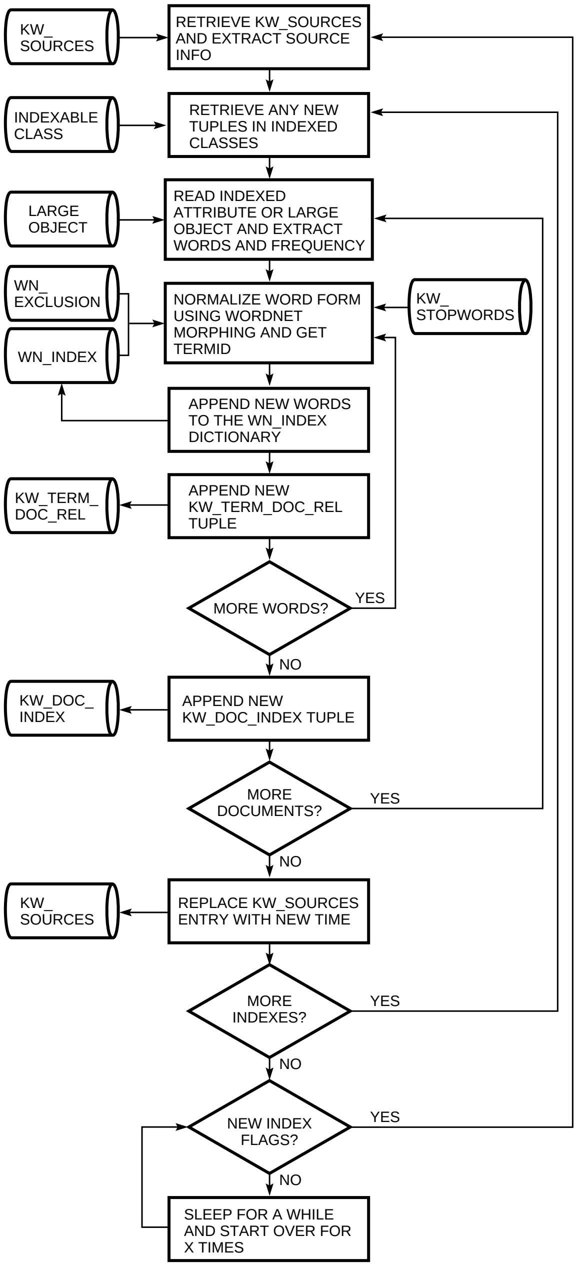

KW_ SOURCES
RETRIEVE KW_SOURCES AND EXTRACT SOURCE INFO
INDEXABLE CLASS
RETRIEVE ANY NEW TUPLES IN INDEXED CLASSES
LARGE OBJECT
READ INDEXED ATTRIBUTE OR LARGE OBJECT AND EXTRACT WORDS AND FREQUENCY
WN_ EXCLUSION
WN_INDEX
NORMALIZE WORD FORM USING WORDNET MORPHING AND GET TERMID
KW_ STOPWORDS
APPEND NEW WORDS TO THE WN_INDEX DICTIONARY
KW_TERM_ DOC_REL
APPEND NEW KW_TERM_DOC_REL TUPLE
MORE WORDS?
YES
NO
KW_DOC_ INDEX
APPEND NEW KW_DOC_INDEX TUPLE
MORE DOCUMENTS?
YES
NO
KW_ SOURCES
REPLACE KW_SOURCES ENTRY WITH NEW TIME
MORE INDEXES?
YES
NO
NEW INDEX FLAGS?
YES
NO
SLEEP FOR A WHILE AND START OVER FOR X TIMES

Figure 3
The Lassen Indexing Daemon Process

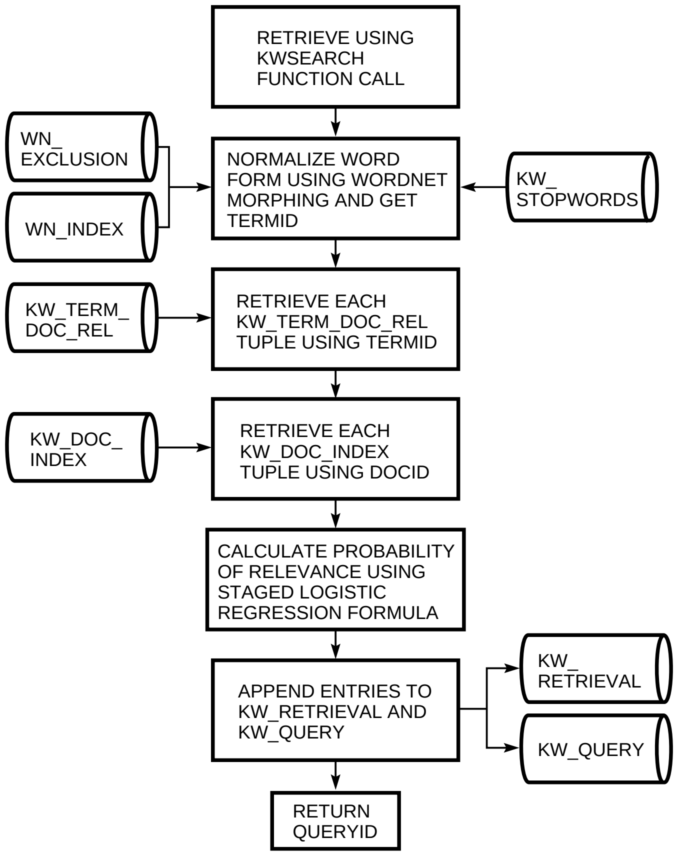

RETRIEVE USING KWSEARCH FUNCTION CALL
WN_ EXCLUSION
WN_INDEX
NORMALIZE WORD FORM USING WORDNET MORPHING AND GET TERMID
KW_ STOPWORDS
KW_TERM_ DOC_REL
RETRIEVE EACH KW_TERM_DOC_REL TUPLE USING TERMID
KW_DOC_ INDEX
RETRIEVE EACH KW_DOC_INDEX TUPLE USING DOCID
CALCULATE PROBABILITY OF RELEVANCE USING STAGED LOGISTIC REGRESSION FORMULA
APPEND ENTRIES TO KW_RETRIEVAL AND KW_QUERY
KW_ RETRIEVAL
KW_QUERY
RETURN QUERYID

Figure 4
The Lassen Retrieval Process

same processing steps as documents in the indexing process. The individual words of the query are extracted and located in the wn_index dictionary (after removing common words or "stopwords"). The termids for matching words from wn_index are then used to retrieve all the tuples in kw_term_doc_rel that contain the term. For each unique document identifier in this list of tuples, the matching kw_doc_index tuple is retrieved. With the frequency information contained in kw_term_doc_rel and kw_doc_index, the estimated probability of relevance is calculated for each document that contains at least one term in common with the query. The formulae used in the calculation are based on experiments with full-text retrieval.[8] The basic equation for the probabilistic model used in Lassen states the following: The probability of the event that a document is relevant R, given that there is a set of N "clues" associated with that document, A_i for $i = 1, 2, \ldots, N$, is

$$\log O(R|A_1,\ldots,A_N) = \log O(R) + \sum_{i=1}^{N}[\log O(R|A_i) - \log O(R)], \qquad (1)$$

where for any events E and E', the odds $O(E|E')$ is $P(E|E')/P(\bar{E}|E')$, i.e., a simple transformation of the probabilities. Because there is not enough information to compute the exact probability of relevance for any user and any document, an estimation is derived based on logistic regression of a set of clues (usually terms or words) contained in some sample of queries and the documents previously judged to be relevant to those queries. For a set of M terms that occur in both a query and a given document, the regression equation is of the form

$$\log O(R|A_1,\ldots,A_M) \approx c_0 + c_1 \cdot f(M)\sum_{1}^{M} X_{m,1} + \cdots + c_K \cdot f(M)\sum_{1}^{M} X_{m,K} + c_{K+1}M + c_{K+2}M^2, \qquad (2)$$

where there are K retrieval variables $X_{m,K}$ used to characterize each term or clue, and the c_i coefficients are constant for a given training set of queries and documents. The coefficients used in the prototype were derived from analysis of full-text documents and queries (with relevance judgments) from the TIPSTER information retrieval test collection.[9] The derivation of this formula is given in "Probabilistic Retrieval Based on Staged Logistic Regression."[8] The full retrieval equation used for the prototype version of retrieval described in this section is

$$\log O(R|A_1,\ldots,A_M) \approx -3.51 + \frac{1}{\sqrt{M}+1}\Big[37.4\sum_{1}^{M} X_{m,1} + 0.330\sum_{1}^{M} X_{m,2} - 0.1937\sum_{1}^{M} X_{m,3}\Big] + 0.0929M, \qquad (3)$$

where

$X_{m,1}$ is the quotient of the number of times the mth term occurs in the query and the sum of the total number of terms in the query plus 35;

$X_{m,2}$ is the logarithm of the quotient arrived at by dividing the number of times the mth term occurs in the document by the sum of the total number of terms in the document plus 80;

$X_{m,3}$ is the logarithm of the quotient arrived at by dividing the number of times the mth term occurs in the database (i.e., in all documents) by the total number of terms in the collection;

M is the number of terms held in common by the query and the document.

Note that the M^2 term called for in Equation 2 was not found to provide any significant difference in the results and was omitted from Equation 3. The constants 35 and 80, which were used in $X_{m,1}$ and $X_{m,2}$, are arbitrary but appear to offer the best results when set to the average size of a query and the average size of a document for the particular database. The sequence of operations performed to calculate the probability of relevance is shown in Figure 5. Note that in the figure, $k1$, …, $k5$ represent the constants of Equation 3.

The probability of relevance is calculated for each document (by converting the logarithmic odds to a probability) and is stored along with a unique query identifier, the document identifier, and some location information in the kw_retrieval class. The query itself

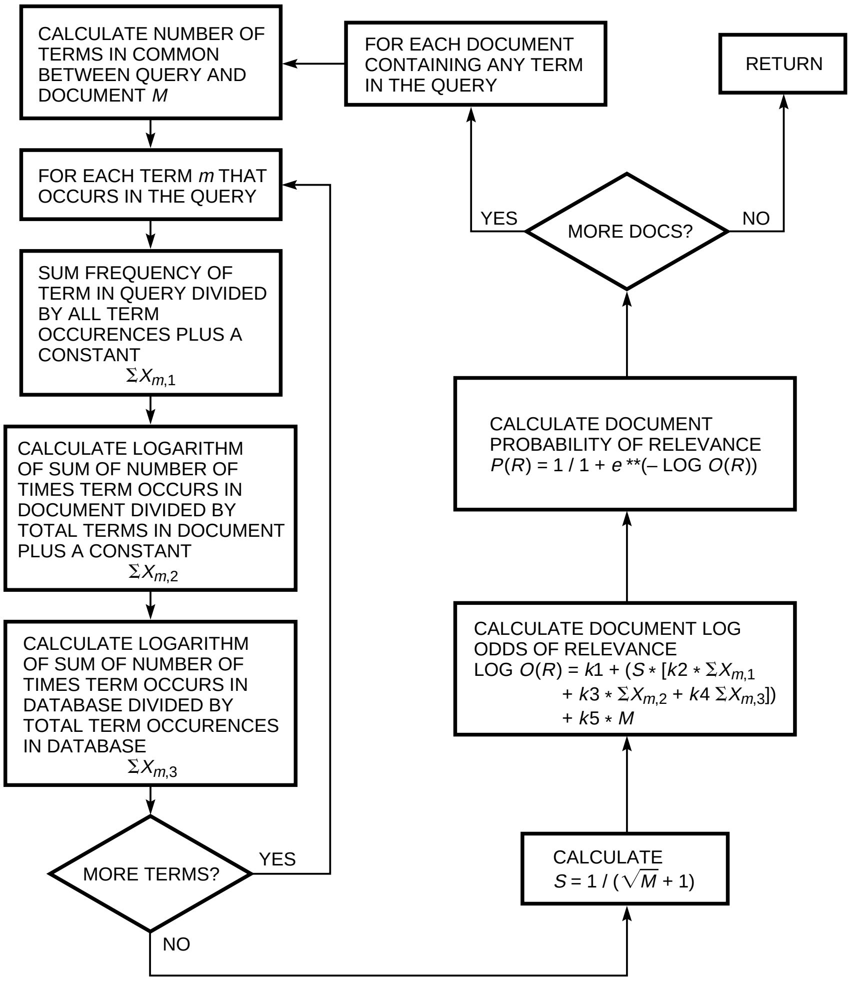

CALCULATE NUMBER OF TERMS IN COMMON BETWEEN QUERY AND DOCUMENT M
FOR EACH DOCUMENT CONTAINING ANY TERM IN THE QUERY
RETURN
FOR EACH TERM m THAT OCCURS IN THE QUERY
YES
MORE DOCS?
NO
SUM FREQUENCY OF TERM IN QUERY DIVIDED BY ALL TERM OCCURENCES PLUS A CONSTANT $\Sigma X_{m,1}$
CALCULATE DOCUMENT PROBABILITY OF RELEVANCE $P(R) = 1 / 1 + e$**(– LOG $O(R)$)
CALCULATE LOGARITHM OF SUM OF NUMBER OF TIMES TERM OCCURS IN DOCUMENT DIVIDED BY TOTAL TERMS IN DOCUMENT PLUS A CONSTANT $\Sigma X_{m,2}$
CALCULATE LOGARITHM OF SUM OF NUMBER OF TIMES TERM OCCURS IN DATABASE DIVIDED BY TOTAL TERM OCCURENCES IN DATABASE $\Sigma X_{m,3}$
CALCULATE DOCUMENT LOG ODDS OF RELEVANCE LOG $O(R) = k1 + (S * [k2 * \Sigma X_{m,1} + k3 * \Sigma X_{m,2} + k4\, \Sigma X_{m,3}]) + k5 * M$
MORE TERMS?
YES
NO
CALCULATE $S = 1 / (\sqrt{M} + 1)$

Figure 5
The Calculation for the Staged Logistic Regression Probabilistic Ranking Process

and its unique identifier are stored in the kw_query class. To see the results of the retrieval operation, the query identifier is used to retrieve the appropriate kw_retrieval tuples, ranked in order according to the estimated probability of relevance. The kw_retrieval and kw_query classes have the following POSTQUEL definitions:

```
create kw_query (
query_id = int4,          /* ID number */
query_user = char16,      /* POSTGRES user name */
query_text = text)        /* the actual query */

create kw_retrieval (
query_id = int4,          /* link to the query */
doc_id = int4,            /* document ID number */
rel_oid = oid,            /* location of doc */
attr_oid = oid,
attr_num = int2,
tuple_id = oid,
doc_len = int4,           /* size of document */
doc_match_terms = int4,   /* number of query terms
                             in the document */
doc_prob_rel = float8)    /* estimated probability
                             of relevance */
```

The algorithm used for ranked retrieval in the Lassen prototype was tested against a number of other systems and algorithms as part of the TREC competition and provided excellent retrieval performance.[10] We have found that the retrieval coefficients used in the formula derived from analysis of the TIPSTER collection appear to work well for a variety of document types. In principle, the staged logistic regression retrieval coefficients should be adapted to the particular characteristics of the database by collecting relevance judgments from actual users and reapplying the staged logistic regression analysis to derive new coefficients. This activity has not been performed for this prototype implementation.

The primary contribution of the Lassen prototype has been as a proof-of-concept for the integration of full-text indexing and ranked retrieval operations in a relational database management system. The prototype implementation that we have described in this section has a number of problems. For example, in the prototype design for indexing and retrieval operations, all the information used is visible in user-accessible classes in the database. Also, the overhead is fairly high, in terms of storage and processing time, for maintaining the indexing and retrieval information in this way. For example, POSTGRES allocates 40 bytes of system information for each tuple in a class, and indexing can take several seconds per document.

Currently, we are investigating a class of new access methods to support indexing and retrieval in a more efficient fashion. The class of methods involves declaring some POSTGRES functions that can extract subelements of a given type of attribute (such as words in a text document) and generate indexes for each of the subelements extracted. Other types of data might also benefit from this class of access methods. For example, functions that extract subelements like geometric shapes from images might be used to generate subelement indexes of image collections. Particular index element extraction methods can be of great value in providing access to the sort of information stored in the Sequoia 2000 Electronic Repository. The following section describes one such index extraction method developed for the special needs of Sequoia 2000 data.

GIPSY: Automatic Georeferencing of Text

Environmental Impact Reports (EIRs), journal articles, technical reports, and myriad other text items related to global change research that might be included in the Sequoia 2000 database are examples of a class of documents that discuss or refer to particular places or regions. A common retrieval task is to find the items that refer to or concentrate on a specific geographic region. Although it is possible to have a human catalog each item for location, one goal of the Electronic Repository was to make all indexing and retrieval automatic, thus eliminating the requirement for human analysis and classification of documents in the database. Therefore, part of our research involved developing methods to perform automatic georeferencing of text documents, that is, to automatically index and retrieve a document according to the geographic locations discussed or displayed in or otherwise associated with its content.

In Lassen and most other full-text information retrieval systems, searches with a geographical component, such as “Find all documents whose contents pertain to location X,” are not supported directly by indexing, query, or display functions. Instead, these searches work only by references to named places, essentially as side effects of keyword indexing. Whereas human indexers are usually able to understand and apply correct references to a document, the costs in time and money of using geographically trained human indexers to read and index the entire contents of a large full-text collection are prohibitive. Even in cases where a document is meticulously indexed manually, geographic index terms consisting of keywords (text strings) have several well-documented problems with ambiguity, synonymy, and name changes over time.[11,12]

Advantages of the GIPSY Model

To deal with these problems, we developed a new model for supporting geographically based access to text.[13] In this model, words and phrases that contain geographic place names or geographic characteristics are extracted from documents and used as input to certain database functions. These functions use spatial reasoning and statistical methods to approximate the geographic position being referenced in the text. The actual index terms assigned to a document are a set of coordinate polygons that describe an area on the earth's surface in a standard geographical projection system. Using coordinates instead of names for the place or geographic characteristic offers a number of advantages.

- Uniqueness. Place names are not unique, e.g., Venice, California, and Venice, Italy, are not apparently different without the qualifying larger region to differentiate them. Using coordinates removes this ambiguity.
- Immunity to spatial boundary changes. Political boundaries change over time, leading to confusion about the precise area being referred to. Coordinates do not depend on political boundaries.
- Immunity to name changes. Geographic names change over time, making it difficult for a user to retrieve all information that has been written about an area during any extended time period. Coordinates remove this ambiguity.
- Immunity to spatial, naming, and spelling variation. Names and terms vary not only over time but also in contemporary usage. Geographic names vary in spelling over time and by language. Areas of interest to the user will often be given place names designated only in the context of a specific document or project. Such variations occur frequently for studies done in oceanic locations. Names associated with these studies are unknown to most users. Coordinates are not subject to these kinds of verbal variations.

Indexing texts and other objects (e.g., photographs, videos, and remote sensing data sets) by coordinates also permits the use of a graphical interface to the information in the database, where representations of the objects are plotted on a map. A map-based graphical interface has several advantages over one that uses text terms or one that simply uses numerical access to coordinates. As Furnas suggests, humans use different cognitive structures for graphical information than for verbal information, and spatial queries cannot be fully simulated by verbal queries.[14] Because many geographical queries are inherently spatial, a graphical model is more intuitive. This is supported by Morris' observation that users given the choice between menu and graphical interfaces to a geographic database preferred the graphical mode.[15] A graphical interface, such as a map, also allows for a dense presentation of information.[16]

To address the needs of global change scientists, the Sequoia 2000 project team proposed a new browser paradigm.[17] This system, called Tioga, displays information topologically according to continuous characteristics that are attributes of the data.[18] For example, documents may be displayed on a map according to their latitude and longitude. Documents may also be displayed according to the time at which they were generated and the time to which they refer, as well as by more abstract functions such as the reading level of the document and the author's attitudes as expressed in the document. A prototype of the geographical browsing component was included in the Lassen Geographic Browser, which is shown in Figure 6.

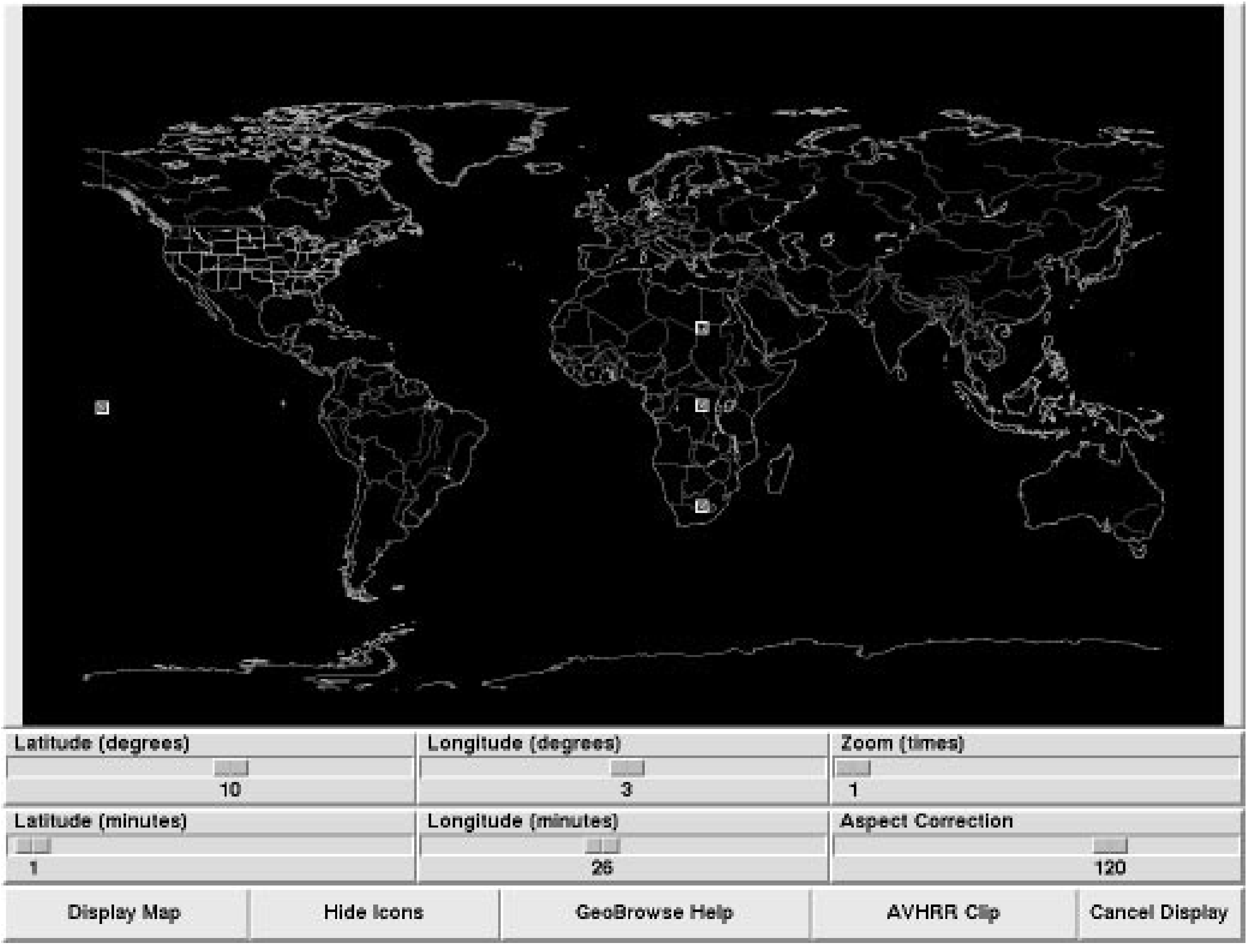
This browser allows any georeferenced object in the database to be indicated by an icon on the map. The user employs the mouse to center the map on any location and to zoom in or out for more or less map detail. Icons can be made to appear at any coordinates and for any range of magnification values. When an icon is selected by the user, a menu of the objects georeferenced at the icon coordinates and detail level are displayed for selection.

An Algorithm to Georeference Text

The advantages of georeferencing are apparent. Not so apparent is how to perform such a task automatically. We developed the following three-part thesaurus-based algorithm to explore this task; the algorithm provides the basis for georeferencing inGIPSY.[19]

1. Identify geographic place names and phrases. This step attempts to recognize all relevant content-bearing geographic words and phrases. The parser for this step must "understand" how to identify geographic terminology of two types:
   a. Terms that match objects or attributes in the data set. This step requires a large thesaurus of geographic names and terms, partially hand built and partially automatically generated.
   b. Lexical constructs that contain spatial information, e.g., "adjacent to the coast," "south of the delta," and "between the river and the highway."

   To implement this part of the algorithm, a list of the most commonly occurring constructs must be created and integrated into a thesaurus.
2. Locate pertinent data. The output of the parser is passed to a function that retrieves geographic coordinate data pertinent to the extracted terms and phrases. Spatially indexed data used in this step can include, for example, name, size, and location of cities and states; name and location of endangered species; and name, location, and bioregional characteristics of different climatic regions. The system must identify the spatial locations that most closely match the geographic terms extracted by the parser and, when geographic modifiers are used, heuristically modify the area of coverage. For example, the phrase "south of Lake Tahoe" will map to the area south of Lake Tahoe, covering approximately the same volume. This spatial representation is, by



Latitude (degrees) 10

Longitude (degrees) 3

Zoom (times) 1

Latitude (minutes) 1

Longitude (minutes) 26

Aspect Correction 120

Figure 6
Screen from the Lassen Geographic Browser

necessity, the result of an arbitrary assumption of size, but its purpose is to provide only partial evidence to be used in determining locations as described below.

Since geopositional data for land use (e.g., cities, schools, and industrial areas) and habitats (e.g., wetlands, rivers, forests, and indigenous species) is also available, extracted keywords and phrases for these types of data must be recognized. The thesaurus entries for this data should incorporate several other types of information, such as synonymy (e.g., Latin and common names of species) and membership (e.g., wetlands contain cattails, but geopositional data on cattails may not exist, so we must use their mention as weak evidence of a discussion of wetlands and use that data instead).

For our implementation ofGIPSY, we used two primary data sets to construct the thesaurus. The first was a subset of the United States Geological Survey's Geographic Names Information System (GNIS).[20] This data set contains latitude/longitude point coordinates associated with over 60,000 geographic place names in California. To facilitate comparison with other data sets, the GNIS latitude/longitude coordinates were converted to the Lambert-Azimuthal projection. Examples of place names with associated points include

University of California Davis: –1867878 –471379

Redding: –1863339 –234894

Data for land use and habitat data was derived in the United States Geological Survey's Geographic Information Retrieval and Analysis System (GIRAS).[21]

Each identified name, phrase, or region description is associated with one or more polygons that may be the place discussed in the text. Weights can be assigned to each of these polygons based on the frequency of use of its associated term or phrase in the text being indexed and in the thesaurus. Many relevant terms do not exactly match place names or the feature and land use types listed above. For example, alfalfa is a crop grown in California and should be associated with the crop data from the GIRAS land use data set. The thesaurus was therefore extended, both manually and by extraction of relationships from the WordNet thesaurus, to include the following types of terms:[7]

synonymy

= : = synonym

kind-of relationships

~ : = hyponym (maple is a ~ of tree)
@ : = hypernym (tree is a @ of maple)

part-of relationships

# : = meronym (finger is a # of hand)
% : = holonym (hand is a % of finger)
& : = evidonym (pine is a & of shortleaf pine)

3. Overlay polygons to estimate approximate locations. The objective of this step is to combine the evidence accumulated in the preceding step and infer a set of polygons that provides a reasonable approximation of the geographical locations mentioned in the text. Each *geophrase, weight, polygon* tuple can be represented as a three-dimensional "extruded" polygon whose base is in the plane of the x- and z-axes and whose height extends upward on the y-axis a distance proportional to its weight (see Figure 7a). As new polygons are added, several cases may arise.
   a. If the base of a polygon being added does not intersect with the base of any other polygons, it is simply laid on the base map beginning at $y = 0$ (see Figure 7b).
   b. If the polygon being added is completely contained within a polygon that already exists on the geopositional skyline, it is laid on top of that extruded polygon, i.e., its base plane is positioned higher on the y-axis (see Figure 7c).
   c. If the polygon being added intersects but is not wholly contained by one or more polygons, the polygon being added is split. The intersecting portion is laid on top of the existing polygon and the nonintersecting portion is positioned at a lower level (i.e., at $y = 0$). To minimize fragmentation in this case, polygons are sorted by size prior to being positioned on the skyline (see Figure 7d).

In effect, the extruded polygons, when laid together, are "summed" by weight to form a geopositional skyline whose peaks approximate the geographical locations being referenced in the text. The geographic coordinates assigned to the text segment being indexed are determined by choosing a threshold of elevation z in the skyline, taking the x-z plane at z, and using the polygons at the selected elevation. Raising the elevation of the threshold will tend to increase the accuracy of the retrieval, whereas lowering the elevation tends to include other similar regions.

To see the results of this process in the GIPSY prototype, consider the following text from a publication of the California Department of Water Resources:

> The proposed project is the construction of a new State Water Project (SWP) facility, the Coastal Branch, Phase II, by the Department of Water Resources (DWR) and a local distribution facility, the Mission Hills Extension, by water purveyors of northern Santa Barbara County. This proposed buried pipeline would deliver 25,000 acre-feet per year (AF/YR) of SWP water to San Luis Obispo County Flood Control and Water Conservation District (SLOCFCWCD) and 27,723 AF/YR to Santa Barbara County Flood Control and Water Conservation District (SBCFCWCD).... This extension would serve the South Coast and Upper Santa Ynez Valley. DWR and the Santa Barbara Water Purveyors Agency are jointly producing an EIR for the Santa Ynez Extension. The Santa Ynez Extension Draft EIR is scheduled for release in spring 1991.[22]

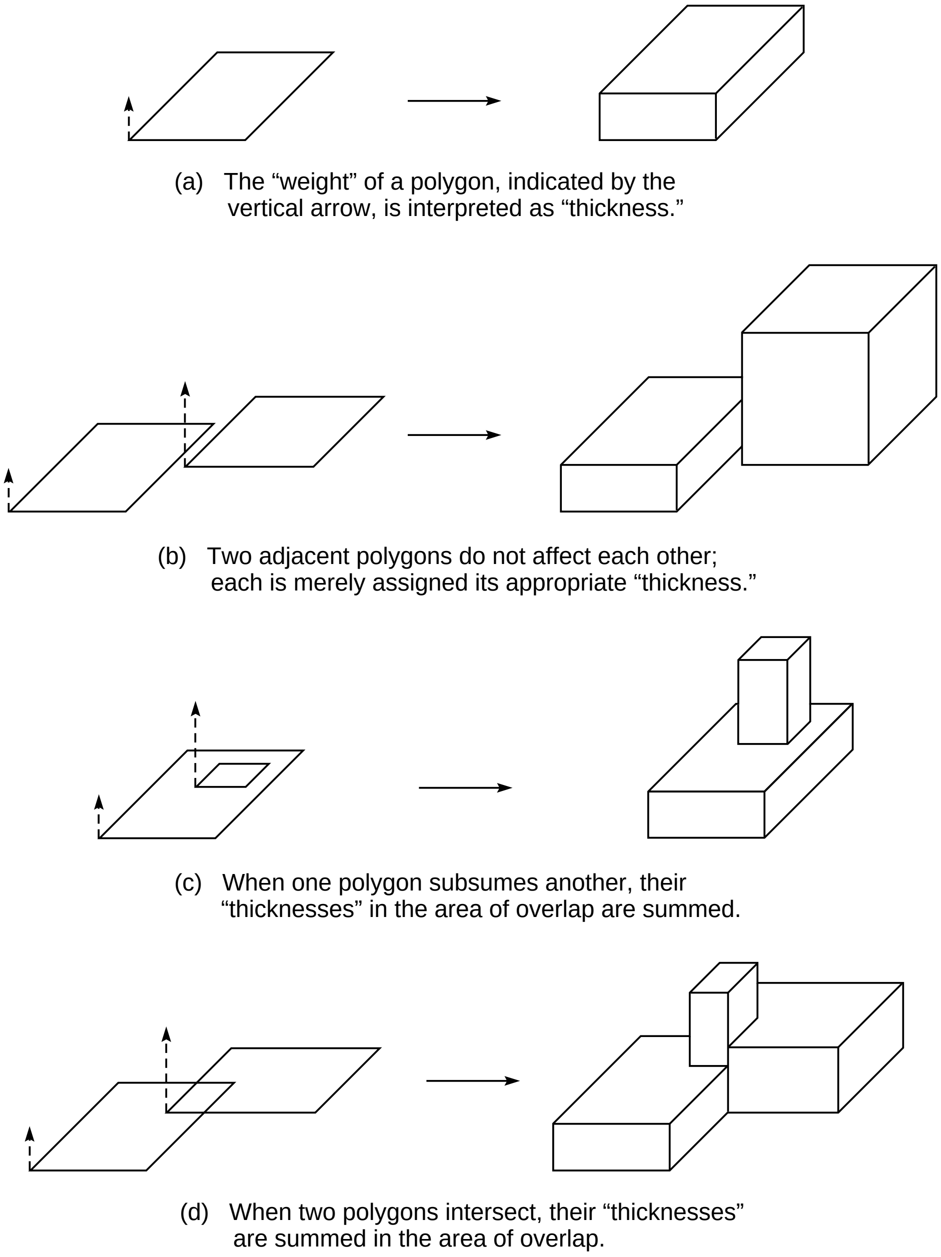
The resulting surface plot appears in Figure 8. The figure contains a gridded representation of the state of California, which is elevated to distinguish it from the base of the grid. The northern part of the state is on the left-hand side of the image. The towers rising over the state's shape represent polygons in the skyline generated by GIPSY's interpretation of the text. The largest towers occur in the area referred to by the text, primarily centered on Santa Barbara County, San Luis Obispo, and the Santa Ynez Valley area.

The surface plots generated in this fashion can also be used for browsing and retrieval. For example, the two-dimensional base of a polygon with a thickness above a certain threshold can be assigned as a coordinate index to a document. These two-dimensional polygons might then be displayed as icons on a map browser such as the one shown in Figure 6.

Future Work

Research remains to be done on several extensions to the existing GIPSY implementation. Because a geographic knowledge base and spatial reasoning are fundamental to the georeferencing process, they have been the focus of initial research efforts.

The existing prototype can be complemented by the addition of more sophisticated natural language processing. For example, spatial reasoning and geographic data could be combined with parsing techniques to develop semantic representations of the text. Adjacency indicators, such as "south of" or "between," should be recognized by a parser. Also, the work on document segmentation described below could be used to explore the locality of reference to geographic entities within full-text documents. GIPSY's technique may be most effective when applied to a paragraph or section level of a text instead of to the entire document.



(a) The “weight” of a polygon, indicated by the vertical arrow, is interpreted as “thickness.”

(b) Two adjacent polygons do not affect each other; each is merely assigned its appropriate “thickness.”

(c) When one polygon subsumes another, their “thicknesses” in the area of overlap are summed.

(d) When two polygons intersect, their “thicknesses” are summed in the area of overlap.

Figure 7
Overlaying Polygons to Estimate Approximate Locations

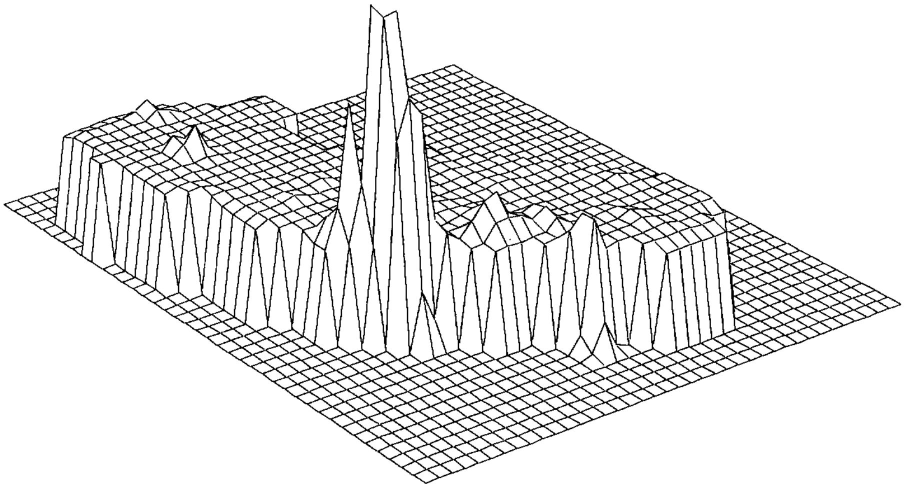


Figure 8
Surface Plot Produced from the State Water Project Text

TextTiling: Enhancing Retrieval through Automatic Subtopic Identification

Full-length documents have only recently become available on-line in large quantities, although technical abstracts, short newswire texts, and legal documents have been accessible for many years.[23] The large majority of on-line information has been bibliographic (e.g., authors, titles, and abstracts) instead of the full text of the document. For this reason, most information retrieval methods are better suited for accessing abstracts than for accessing longer documents. Part of the repository research was an exploration of new approaches to information retrieval particularly suited to full-length texts, such as those expected in the Sequoia 2000 database.

A problem with applying traditional information retrieval methods to full-length text documents is that the structure of full-length documents is quite different from that of abstracts. (In this paper, "full-length document" refers to expository text of any length. Typical examples are a short magazine article and a 50-page technical report. We exclude documents composed of headlines, short advertisements, and any other disjointed texts of whatever length. We also assume that the document does not have detailed orthographically marked structure. Croft, Krovetz, and Turtle describe work that takes advantage of this kind of information.[24]) One way to view an expository text is as a sequence of subtopics set against a backdrop of one or two main topics. A long text comprises many different subtopics that may be related to one another and to the backdrop in many different ways. The main topics of a text are discussed in its abstract, if one exists, but subtopics are usually not mentioned. Therefore, instead of querying against the entire content of a document, a user should be able to issue a query about a coherent subpart, or subtopic, of a full-length document, and that subtopic should be specifiable with respect to the document's main topic(s).

Consider a *Discover* magazine article about the Magellan space probe's exploration of Venus.[25] A reader divided this 23-paragraph article into the following segments with the labels shown, where the numbers indicate paragraph numbers:

1–2	Intro to Magellan space probe
3–4	Intro to Venus
5–7	Lack of craters
8–11	Evidence of volcanic action
12–15	River Styx
16–18	Crustal spreading
19–21	Recent volcanism
22–23	Future of Magellan

Assume that the topic of volcanic activity is of interest to a user. Crucial to a system's decision to retrieve this document is the knowledge that a dense discussion of volcanic activity, rather than a passing reference, appears. Since volcanism is not one of the text's two main topics, the number of references to this term will probably not dominate the statistics of term frequency. On the other hand, document selection should not necessarily be based on the number of references to the target terms.

The goal should be to determine whether or not a relevant discussion of a concept or topic appears. A simple approach to distinguishing between a true discussion and a passing reference is to determine the locality of the references. In the computer science operating systems literature, locality refers to the fact that over time, memory access patterns tend to concentrate in localized clusters rather than be distributed evenly throughout memory. Similarly, in full-length texts, the close proximity of members of a set of references to a particular concept is a good indicator of topicality. For example, the term *volcanism* occurs 5 times in the Magellan article, the first four instances of which occur in four adjacent paragraphs, along with accompanying discussion. In contrast, the term*scientists,* which is not a valid subtopic, occurs 13 times, distributed somewhat evenly throughout. By its very nature, a subtopic will not be discussed throughout an entire text. Similarly, true subtopics are not indicated by only passing references. The term *belly dancer* occurs only once, and its related terms are confined to the one sentence it appears in. As its usage is only a passing reference, belly dancing is not a true subtopic of this text.

Our solution to the problem of retaining valid subtopical discussions while at the same time avoiding being fooled by passing references is to make use of locality information and to partition documents according to their subtopical structure. This approach's capacity for improving a standard information retrieval task has been verified by information retrieval experiments using full-text test collections from the TIPSTER database.[26,27]

One way to get an approximation of the subtopic structure is to break the document into paragraphs, or for very long documents, sections. In both cases, this entails using the orthographic marking supplied by the author to determine topic boundaries.

Another way to approximate local structure in long documents is to divide the documents into even-sized pieces, without regard for any boundaries. This approach is not practical, however, because we are interested in exploring the performance of motivated segmentation, i.e., segmentation that reflects the text's true underlying subtopic structure, which often spans paragraph boundaries.

Toward this end, we have developed TextTiling, a method for partitioning full-length text documents into coherent multiparagraph units called tiles.[26,28,29] TextTiling approximates the subtopic structure of a document by using patterns of lexical connectivity to find coherent subdiscussions. The layout of the tiles is meant to reflect the pattern of subtopics contained in an expository text. The approach uses quantitative lexical analyses to determine the extent of the tiles and to classify them with respect to a general knowledge base. The tiles have been found to correspond well to human judgments of the major subtopic boundaries of science magazine articles.

The algorithm is a two-step process. First, all pairs of adjacent blocks of text (where blocks are usually three to five sentences long) are compared and assigned a similarity value. Second, the resulting sequence of similarity values, after being graphed and smoothed, is examined for peaks and valleys. High similarity values, which imply that the adjacent blocks cohere well, tend to form peaks, whereas low similarity values, which indicate a potential boundary between tiles, create valleys. Figure 9 shows such a graph for the *Discover* magazine article mentioned earlier. The vertical lines indicate where human judges thought the topic boundaries should be placed. The graph shows the computed similarity of adjacent blocks of text. Peaks indicate coherency, and valleys indicate potential breaks between tiles.

The one adjustable parameter is the size of the block used for comparison. This value, *k,* varies slightly from text to text. As a heuristic, it is assigned the average paragraph length (in sentences), although the block size that best matches the human judgment data is sometimes one sentence greater or smaller. Actual paragraphs are not used because their lengths can be highly irregular, leading to unbalanced comparisons.

Similarity is measured by using a variation of the tf.idf (term frequency times inverse document frequency) measurement.[30] In standard tf.idf, terms that are frequent in an individual document but relatively infrequent throughout the corpus are considered to be good distinguishers of the contents of the individual document. In TextTiling, each block of *k* sentences is treated as a unit, and the frequency of a term within each block is compared to its frequency in the entire document. (Note that the algorithm uses a large stop list; i.e., closed class words and other very frequent terms are omitted from the calculation.) This approach helps bring out a distinction between local and global extent of terms. A term that is discussed frequently within a localized cluster (thus indicating a cohesive passage) will be weighted more heavily than a term that appears frequently but scattered evenly throughout the entire document, or infrequently within one block. Thus if adjacent blocks share many terms, and those shared terms are weighted heavily, there is strong evidence that the adjacent blocks cohere with one another.

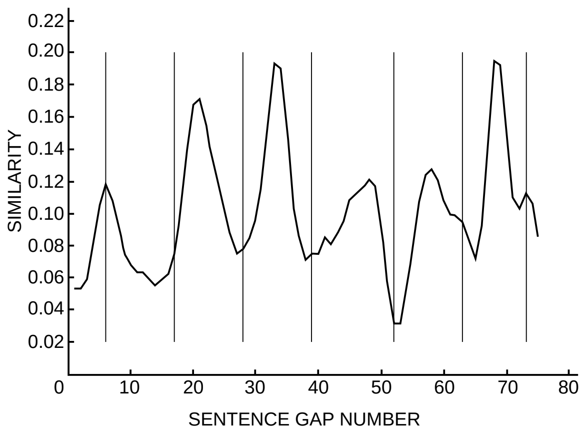


Figure 9
Results of TextTiling a 77-sentence Science Article

Similarity between blocks is calculated by the following cosine measure: Given two text blocks $b1$ and $b2$,

$$\cos(b1,b2) = \frac{\sum_{t=1}^{n} w_{t,b1}\, w_{t,b2}}{\sqrt{\sum_{t=1}^{n} w_{t,b1}^2 \sum_{t=1}^{n} w_{t,b2}^2}},$$

where t ranges over all the terms in the document, and $w_{t,b1}$ is the tf.idf weight assigned to term t in block $b1$. Thus, if the similarity score between two blocks is high, then not only do the blocks have terms in common, but the common terms are relatively rare with respect to the rest of the document. The evidence in the reverse is not as conclusive. If adjacent blocks have a low similarity measure, this does not necessarily mean that the blocks cohere. In practice, however, this negative evidence is often justified.

The graph is then smoothed using a discrete convolution[31] of the similarity function with the function $h_k(.)$, where

$$h_k(i) \equiv \begin{cases} \frac{1}{k^2}(k - |i|), & |i| \leq k - 1 \\ 0, & \text{otherwise.} \end{cases}$$

The result is smoothed further with a simple median smoothing algorithm to eliminate small local minima.[32] Tile boundaries are determined by locating the lowermost portions of valleys in the resulting plot. The actual values of the similarity measures are not taken into account; the relative differences are what are of consequence.

Retrieval processing should reflect the assumption that full-length text is meaningfully different in structure from abstracts and short articles. We have conducted retrieval experiments that demonstrate that taking text structure into account can produce better results than using full-length documents in the standard way.[26,28,29] By working within this paradigm, we have developed an approach to vector-space-based retrieval that appears to work better than retrieving against entire documents or against segments or paragraphs alone.

The resulting retrieval method matches a query against motivated segments and then sums the scores from the top segments for each document. The highest resulting sums indicate which documents should be retrieved. In our test set, this method produced higher precision and recall than retrieving against entire documents or against segments or paragraphs alone.[26] Although the vector-space model of retrieval was used for these experiments, probabilistic models such as the one used in Lassen are equally applicable, and the method should provide similar improvement in retrieval performance.

We believe that recognizing the structure of full-length text for the purposes of information retrieval is very important and will produce considerable improvement in retrieval effectiveness over most existing similarity-based techniques.

Conclusion

The Sequoia 2000 Electronic Repository project has provided a test bed for developing and evaluating technologies required for effective and efficient access to the digital libraries of the future. We can expect that as digital libraries proliferate and include vast databases of information linked together by high-bandwidth networks, they must support all current and future media in an easily accessible and content-addressable fashion.

The work begun on the Sequoia 2000 Electronic Repository is continuing under UC Berkeley's digital library project sponsored jointly by the National Science Foundation (NSF), the National Aeronautics and Space Administration (NASA), and the Defense Advanced Research Projects Agency (DARPA). Digital libraries are a fledgling technology with no firm standards, architectures, or even consensus notions of what they are and how they are to work. Our goal in this ongoing research is to develop the means of placing the contents of this developing global virtual library at the fingertips of a worldwide clientele. Achieving this goal will require the application of advanced techniques for information retrieval, information filtering, resource discovery, and the application of new techniques for automatically analyzing and characterizing data sources ranging from texts to videos. Much of the work needed to enable our vision of these new technologies was pioneered in the Sequoia 2000 Electronic Repository project.

References

1. J. Dozier, "How Sequoia 2000 Addresses Issues in Data and Information Systems for Global Change," Sequoia 2000 Technical Report 92/14 (S2K-92-14) (Berkeley, Calif.: University of California, Berkeley, 1992) (ftp://s2k-ftp.cs.berkeley.edu/pub/sequoia/tech-reports/s2k-9 2-14/s2k-92-14.ps).

2. M. Stonebraker, "An Overview of the Sequoia 2000 Project," *Digital Technical Journal,* vol. 7, no. 3 (1995, this issue): 39–49.

3. M. Stonebraker and G. Kemnitz, "The POSTGRES Next-generation Database Management System," *Communications of the ACM,* vol. 34, no. 10 (1991): 78–92.

4. N. Fuhr, "A Probabilistic Relational Model for the Integration of IR and Databases," *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '93),* Pittsburgh, June 27–July 1, 1993 (New York: Association for Computing Machinery, 1993): 309–317.

5. D. Blair, "An Extended Relational Document Retrieval Model," *Information Processing and Management,* vol. 24 (1988): 349–371.

6. R. Larson, "Design and Development of a Network-Based Electronic Library," *Navigating the Networks: Proceedings of the ASIS Midyear Meeting,* Portland, Oregon, May 21–25, 1994 (Medford, N.J.: Learned Information, Inc., 1994): 95–114. Also available as Sequoia 2000 Technical Report 94/54, July 1994 (ftp://s2k-ftp.cs.berkeley.edu/pub/sequoia/tech-reports/s2k-9 4-54/s2k-94-54.ps).

7. G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, "Five Papers on WordNet," CSL Report 43 (Princeton, N.J.: Princeton University: Cognitive Science Laboratory, 1990).

8. W. Cooper, F. Gey, and D. Dabney, "Probabilistic Retrieval Based on Staged Logistic Regression," *Proceedings of the Fifteenth Annual InternationalACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '92),* Copenhagen, Denmark, June 21–24, 1992 (New York: Association for Computing Machinery, 1992): 198–210.

9. D. Harman, "The DARPA TIPSTER Project," *SIGIR Forum,* vol. 26, no. 2 (1992): 26–28.

10. W. Cooper, A. Chen, and F. Gey, "Experiments in the Probabilistic Retrieval of Full Text Documents," *Text Retrieval Conference (TREC-3) Draft Conference Papers* (Gaithersburg, Md.: National Institute of Standards and Technology, 1994).

11. A. Griffiths, "SAGIS: A Proposal for a Sardinian Geographical Information System and an Assessment of Alternative Implementation Strategies," *Journal of Information Science,* vol. 15 (1989): 261–267.

12. D. Holmes, "Computers and Geographic Information Access," *Meridian,* vol. 4 (1990): 37–49.

13. A. Woodruff and C. Plaunt, "GIPSY: Georeferenced Information Processing SYstem," *Journal of the American Society for Information Science,* vol. 45, no. 9 (1994): 645–655.

14. G. Furnas, "New Graphic Reasoning Models for Understanding Graphical Interfaces," *Human Factors in Computing Systems: Reaching Through Technology Proceedings (CHI '91 Conference),* New Orleans, April-May 1991 (New York: Association for Computing Machinery, 1991): 71–78.

15. B. Morris, "CARTO-NET: Graphic Retrieval and Management in an Automated Map Library," *Special Libraries Association, Geography and Map Division Bulletin,* vol. 152 (1988): 19–35.

16. C. McCann, M. Taylor, and M. Tuori, "ISIS: The Interactive Spatial Information System," *International Journal of Man-Machine Studies,* vol. 28 (1988): 101–138.

17. J. Chen, R. Larson, and M. Stonebraker, "Sequoia 2000 Object Browser," *Digest of Papers, Thirty-seventh IEEE Computer Society International Conference (COMPCON Spring 1992),* San Francisco, February 24-28, 1992 (Los Alamitos, Calif.: Computer Society Press, February 1992): 389–394.

18. M. Stonebraker, J. Chen, N. Nathan, C. Paxson, and J. Wu, "Tioga: Providing Data Management Support for Scientific Visualization Applications," *Proceedings of the Nineteenth International Conference on Very Large Data Bases,* Dublin, Ireland (August 1993): 25–38.

19. A. Woodruff and C. Plaunt, "Automated Geographic Indexing of Text Documents," Sequoia 2000 Technical Report 94/41 (S2K-94-41) (Berkeley, Calif.: University of California, Berkeley, 1994) (ftp://s2k-ftp.cs.berkeley.edu/pub/sequoia/tech-reports/s2k-9 4-41/s2k-94-41.ps).

20. Geographic Names Information System/United States Department of the Interior, United States Geological Survey, rev. ed., Data User's Guide, vol. 6 (Reston, Va.: United States Geological Survey, 1987).

21. J. Anderson, E. Hardy, J. Roach, and R. Witmer, "A Land Use and Land Cover Classification System for Use with Remote Sensor Data," United States Geological Survey Professional Paper #964 (Washington,D.C.: United States Government Printing Office, 1976).

22. State Water Project, Coastal Branch, Phase II, and Mission Hills Extension (Sacramento, Calif.: California Department of Water Resources, 1991).

23. C. Tenopir and J. Ro, *Full Text Databases*(New York: Greenwood Press, 1990).

24. W. Croft, R. Krovetz, and H. Turtle, "Interactive Retrieval of Complex Documents," *Information Processing and Management,* vol. 26, no. 5 (1990): 593–616.

25. A. Chaikin, "Magellan Pierces the Venusian Veil," *Discover,* vol. 13, no. 1 (January 1992).

26. M. Hearst and C. Plaunt, "Subtopic Structuring for Full-Length Document Access," *Proceedings of the Sixteenth Annual InternationalACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '93),* Pittsburgh, June 1993 (New York: Association for Computing Machinery, 1993): 59–68.

27. M. Hearst, "Context and Structure in Automated Full-Text Information Access," Ph.D. dissertation, Report No. UCB/CSD-94/836 (Berkeley, Calif.: University of California, Berkeley, Computer Science Division, 1994).

28. M. Hearst, "TextTiling: A Quantitative Approach to Discourse Segmentation," Sequoia 2000 Technical Report 93/24 (S2K-93-24) (Berkeley, Calif.: University of California, Berkeley, 1993) (ftp://s2k-ftp.cs.berkeley.edu/pub/sequoia/tech-reports/s2k-9 3-24/s2k-93-24.ps).

29. M. Hearst, "Multi-Paragraph Segmentation of Expository Text," *Proceedings of the Thirty-second Meeting of the Association for Computational Linguistics,* Los Cruces, New Mexico, June 1994.

30. G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer* (Reading, Mass.: Addison-Wesley, 1989).

31. The authors are grateful to Michael Braverman for proving that the smoothing algorithm is equivalent to this convolution.

32. L. Rabiner and R. Schafer, *Digital Processing of Speech Signals* (Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1978).

Biographies

Ray R. Larson
Ray Larson is an Associate Professor at the University of California, Berkeley, in the School of Information Management and Systems (formerly the School of Library and Information Studies). He teaches courses and conducts research on the design and evaluation of information retrieval systems. Ray received his Ph.D. from the University of California. He is a member of the American Society for Information Science (ASIS), the Association for Computing Machinery (ACM), the IEEE Computer Society, the American Association for the Advancement of Science, and the American Library Association. He is the Associate Editor for *ACM Transactions on Information Systems* and received the *ASIS Journal* Best Paper Award in 1993.

Christian Plaunt
Christian Plaunt is a doctoral student and graduate research assistant at the University of California, Berkeley, School of Information Management and Systems. His interests include experimental information retrieval system modeling, simulation, design, and evaluation; artificial intelligence techniques for information retrieval; multistage retrieval techniques; information filtering; and music. Chris holds master's degrees in library and information studies and in music (composition). In his spare time, he composes, plays the piano, and works in the Music Library at California State University, Fresno, near which he lives with his wife and their three Siamese cats.

Allison G. Woodruff
Allison Woodruff is a Ph.D. student in the Electrical Engineering and Computer Science Department at the University of California, Berkeley. Her research interests include spatial information systems, multimedia databases, visual programming languages, and user interfaces. Previously, she worked as a geographic information systems specialist for the California Department of Water Resources. Allison holds a B.A. in English from California State University, Chico, and an M.A. in linguistics and an M.S. in computer science from the University of California, Davis.

Marti A. Hearst
Currently a member of the research staff at Xerox Palo Alto Research Center, Marti Hearst completed her Ph.D. in computer science at the University of California, Berkeley, in April 1994. Her dissertation examined context and structure of full-text documents for information access. Her current research interests include intelligent information access, corpus-based computational linguistics, user interfaces, and psycholinguistics.