When Big Data Leads To Lost Data

V.M. Megler
Portland State University
Department of Computer Science
Portland, Oregon
vmegler@cs.pdx.edu

David Maier
Portland State University
Department of Computer Science
Portland, Oregon
maier@cs.pdx.edu

ABSTRACT
For decades, scientists bemoaned the scarcity of observational data to analyze and against which to test their models. Exponential growth in data volumes from ever-cheaper environmental sensors has provided scientists with the answer to their prayers: "big data". Now, scientists face a new challenge: with terabytes, petabytes or exabytes of data at hand, stored in thousands of heterogeneous datasets, how can scientists find the datasets most relevant to their research interests? If they cannot find the data, then they may as well never have collected it; that data is lost to them. Our research addresses this challenge, using an existing scientific archive as our test-bed. We approach this problem in a new way: by adapting Information Retrieval techniques, developed for searching text documents, into the world of (primarily numeric) scientific data. We propose an approach that uses a blend of automated and "semi-curated" methods to extract metadata from large archives of scientific data. We then perform searches over the extracted metadata, returning results ranked by similarity to the query terms. We briefly describe an implementation performed at an ocean observatory to validate the proposed approach. We propose performance and scalability research to explore how continued archive growth will affect our goal of interactive response, no matter the scale.

Categories and Subject Descriptors
H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing – *abstracting methods.* H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *retrieval models, search process.* H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *performance evaluation (efficiency and effectiveness).* H.2.8 [**Information Systems**]: Database Applications – *scientific databases, spatial databases & GIS.*

General Terms
Design, Performance

Keywords
Scientific data, ranked data search.

1. INTRODUCTION
The exponential growth of data from sensors – "the data deluge" [16] that has grown into "big data" – is now old news. New big-data proposals are announced daily, and Gartner [11] warns of big-data chaos if new management techniques are not developed to access the variety, velocity and volume of data now available.

PIKM'12, November 2, 2012, Maui, Hawaii, USA.

In our work with an ocean observatory, CMOP [6], we see the effect this growth has on scientists. Microbiologists, who once collected a few water samples a year and studied them intensively, now have equipment that can capture a water sample every few seconds; on science cruises, sensors measure environmental variables every few milliseconds over hundreds of miles. The methods available for managing and exploiting the information now being collected have not kept pace. One problem of particular concern to our scientists is the following: in their large and growing archive with thousands of datasets, how can they quickly find relevant data? Anecdotal evidence indicates that even when a scientist has previously worked with a dataset, he or she may "misremember" exactly what time or place it covered. Given current access methods, such a dataset is effectively "lost", as far as that scientist is concerned. What they desire is a "Google for data". In this paper, we describe our first steps towards giving the scientists such a tool.

The archive of scientific data of primary interest to our scientists is an observational archive, many terabytes in size, with a collection of data spanning more than a decade [22]. The observation platforms are both fixed and mobile; the observations may be as frequent as every few milliseconds, or as rare as a single, complex sample. Each observation consists of a time, a geospatial location, and a set of environmental variables; related observations are stored together in datasets. The environmental variables (hereafter called variables) that are observed differ over time; there are frequent changes in the instruments deployed as new instruments are developed and new research topics studied. The scientists analyze historical observations and create complex simulation models, in essence creating further (human-made) observations. The datasets are heterogeneous in contents, format, storage type and storage location; a different tool must be used to access and read each dataset type. There are billions (soon to be trillions) of observations; re-processing all existing data into a common format is not feasible. In addition, some of the desired data may exist in other, partner archives rather than their own, adding to the complexity.

Scientists are frustrated by the database-style query and structured searches currently available to them for locating data. If they are not aware of exactly what data exists and which tool to use to access it, or they misremember the exact combination of query terms or selections to find the desired data, they may find no data. Other query terms or selections may result in more datasets than they can review or process. This process often leads to a time-consuming process of repeated searches. When each search requires many selections or scanning a large archive, scientists may desist, at cost to their research. The scientists at CMOP brought this issue of finding relevant data to our attention as one of their highest priority problems with their computing infrastructure (CMOP RIG meeting, July 15, 2010, private communication).

We address their problem by adapting techniques from the field of Information Retrieval (IR) to searching over scientific data. We propose a method for summarizing each dataset and its contents,

and a method for searching over the summaries thus created; our high-level architecture, adapted from well-known Internet search techniques, is shown in Figure 1.

In text-retrieval systems, interactive search is performed by first asynchronously scanning the source text documents in order to extract features; often the features extracted are the list of words used in the document and the frequencies of their use. The list is stored in an inverted index. During a search the user's query terms are compared to the index, and a "similarity measure" applied to quantify the gap between the query terms and each document's index entries. The similarity measure is often a Euclidean distance calculation between a vector representing the query terms and a vector representing each document's features. Documents may be ranked based on increasing values of the distance measure [23]. Research shows that 95% of users would rather have an approximate answer than none at all [8].

Analogously, if it were possible to extract features from each dataset (or part thereof) in a scientific archive and summarize those features into some kind of index, and to then find a method for searching over those features and ranking them with respect to their closeness (however defined) to a scientist's query, it might be possible to provide a set of ranked datasets in response to a query over the archive. The ranking of each dataset should be based on a relevance score that represents an estimate of the dataset content's relevance to the scientist's query terms.

The first author's dissertation will explore these ideas. It will describe the motivation of this work; expand upon our published work on dataset similarity and metadata extract; describe our prototype design and architecture [23]; and report on completed user validation studies. Extension of the description of our model and the planned performance work presented in this paper will complete the dissertation.

Section 2 of this paper gives an overview of a prototype our scientists are now using. We describe our model in Section 3. Section 4 gives a brief summary of two user studies that show our approach resonates with our user community. We describe planned performance and scalability work in Section 5, comment on related work in Section 6 and conclude in Section 7.

2. THE PROTOTYPE

Our prototype [22, 24] consists of three main components, following the architecture shown in Figure 1: a suite of offline feature extraction programs that create our metadata catalog; a scoring and ranking component (also called the search engine); and a user interface. The prototype is now in use by our scientists; after some period of validation it will be made available to external users.

Dataset id:	**saturn01.ctd.201005**
Description:	Saturn-01 Profiler, May 2010
Quality:	Verified
Times [start .. end]:	2010-05-14 .. 2010-05-31
Geometry (location):	Point(-123.87,46.23)
Elevations, datum:	-13 .. 2.5 [m], NGVD27
# Observations:	247,377
Data Location:	http://...
Data Format:	NetCDF
Variables [units] (values):	Salinity [psu] (0 .. 29.6) Temperature [C] (8.2 .. 14.6) Time [secs since epoch] (1,273,869,578 .. 1,275,378,800)

Figure 2. Example Dataset Metadata Record

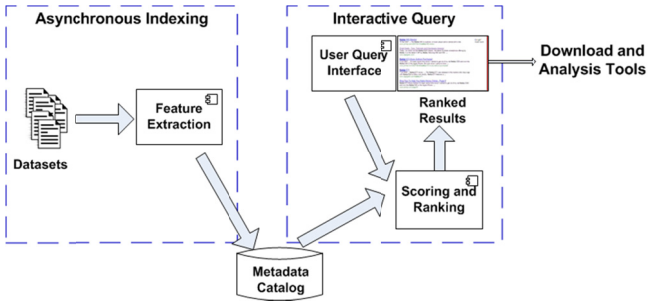

Figure 1. High-Level Architecture for Dataset Similarity

We built a suite of feature extraction programs, one for each different type of data. An extract program summarizes a given dataset once, performing any customized processing required for its data. We build our metadata by processing context given to us (e.g., agency name, data quality), facts automatically extracted from the system (e.g., filename, date last updated), and the contents of the dataset in its native format (e.g., variable names and values; units if available in the dataset header). For example, a dataset with millions of observations taken during a multi-week science cruise may be stored in a relational database table (or alternatively, as a NetCDF file). The data curator tells us the table to process, and to process it as a cruise. We identify the columns involved from the database catalog (or file header) and use those as the variable names (unless given other metadata to map column names to variable names); summarize the millions of individual physical locations of the observations as a simplified polyline representing the cruise path; and split the polyline into multiple line segments. We identify start and end times, variables collected, their numeric values and counts for each line segment. We then build hierarchically linked catalog records for the entire cruise, for each day and each separate line segment; thus, a single catalog record may represent a subset of table rows. A stationary observing platform does not require this processing; however, data collected for over a decade may be stored in a hierarchy of directories with a separate dataset for each month and instrument. An example of such a catalog record is shown in Figure 2.

Our model is "semi-curated" data, minimizing the human intervention required. In general, the data curator must configure or code certain options once for each type of data cataloged; these options are used in generating catalog entries for additional datasets of the same type. Once an extractor has been built for a specific kind of data, new datasets of the same type can be processed automatically. All possible complexity is handled during extraction. Since we read the data only once, extraction processing speed is not critical.

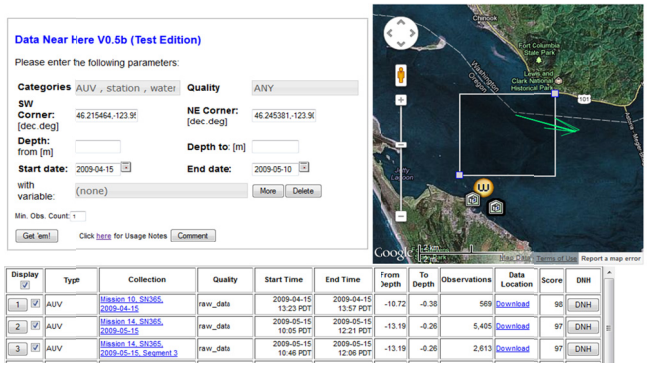

Figure 3. The Data Near Here query interface with ranked search results. Results outside the query area are returned.

As soon as a catalog record is created in the metadata catalog, it is available for searching. The user interface is shown in Figure 3. A user can define query terms for some or all of the following: geographic area (either on the map or by entering geographic coordinates); depth; date range; and one or more variables. She can request "variable existence", that is, datasets for which that variable has been collected; or a range of variable values, specified in one of the available units [22]. The search engine returns ranked results which are displayed on the map, with dataset summaries displayed below the query interface. Datasets are ranked by similarity to the query, using a simple distance measure [24].

3. MODEL AND APPROACH

In this section, we describe our model for datasets and their representation as summaries; how we express a scientific information need as a set of query conditions; and our approach to applying a similarity measure for comparing query terms to the summaries.

3.1 Model

We use as an example a scientist searching for observations taken in mid-2010 in a specific area, containing temperature data with values in the range of 5 to 10C. There are tens of thousands of datasets, and many of them are large; a single dataset may contain millions of records. Many of the datasets may cover a place or time not relevant to the scientist's information need, or may not contain temperature data; those that do, may not contain data in the desired range for that variable. It is not practical to scan the entire archive of datasets and their contents, while still giving real-time response to a query. Therefore, in common with most IR systems [28], we operate on a summary of each dataset, rather than on the dataset itself. A collection of dataset summaries makes up our metadata catalog.

We refer to an example dataset, *saturn01.ctd.201005*, described in Figure 2, containing temperature and salinity data over a one-month period. This example dataset contains many temperature values within the desired range, without containing the actual values of 5 or 10. We also have other datasets containing temperature values in the range 10.01 to 15C taken in late 2010; we would like to find those datasets in preference to those, say, from 2008 with temperatures between 20 and 25C. We desire a notion of "closeness" that we can apply at a dataset level that is more nuanced than "same", "intersects", or "contains this term".

Formally: we wish to relate a dataset d to a query Q via a similarity function $Sim(Q, d)$ that quantifies the similarity between the two. However, evaluating a similarity function over many large datasets at query time is unlikely to yield interactive response times. Thus, for each dataset d we wish to find a small summary s of d that we can use to compute similarity more quickly. Hence, we desire a summarization function F that operates over d to produce a summary s, and a similarity function $Sim\_s(Q, s)$ that produces (approximately) the same results as $Sim(Q, d)$:

$$s = F(d) \tag{1}$$

$$Sim(Q, d) \approx Sim\_s(Q, s) = Sim\_s(Q, F(d)) \tag{2}$$

Our strategy is to choose summarization and similarity functions that allow us to compute the summary s in advance, and evaluate $Sim\_s(Q, s)$ quickly at the time query Q is presented.

Creating a Dataset Summary. First, we describe the dataset summarization function. We begin with a collection of n datasets $D = \{d_1, d_2, \ldots, d_n\}$. We consider each dataset d_i as consisting of some "global" information d_g that applies to the whole dataset, such as identifier, data location and format; and a set of m columns $\{c_1, c_2, \ldots, c_m\}$. In our example dataset in Figure 2, the set of columns is $\{salinity, temperature, time\}$.

We create the summary $s_i = F(d_i)$ from each dataset d_i by applying a summarization function F. As is common in text and image retrieval, the summary will be a list of p features, each produced by a function $f_i(d)$. Thus, the summary is of the form:

$$s_i = F(d_i) = \langle f_1(d_i), f_2(d_i), \ldots, f_p(d_i) \rangle \tag{3}$$

Each function f_i summarizes a particular aspect of the dataset. There are many possible such functions. To maximize utility, we wish to identify functions that capture features that most closely match the way our scientists appear to think about their data. Further, we wish to capture features that correspond to possible query terms. We conjecture that scientists often visualize their data in tables with each variable in a separate column; and that their information needs can, in general, be fairly arbitrary subsets of the data in these columns. Therefore, in order to match possible query terms to captured features, we generally summarize each column separately. However, where a group of columns has semantic meaning (such as a latitude column and a longitude column, together defining geospatial location), we may summarize the group with a single feature. We may choose to do both: summarize a group of columns to produce a single feature in addition to summarizing each of the columns as individual features.

For simplicity we refer to the summary for column x as $sc_x = f_x(d[c_x])$, where $d[c_x]$ is the x^{th} column of dataset d and the summary sc_x is produced by the function f_x. We capture dataset-level summary information sg for the dataset using a function $sg = f_g(d)$. Thus, a dataset summary is described by:

$$s = \langle sg, sc_1, sc_2, \ldots, sc_m \rangle \tag{4}$$

$$= \langle f_g(d), f_1(d[c_1]), f_2(d[c_2]), \ldots, f_m(d[c_m]) \rangle$$

However, note that we may also have $sc_x = f_x(d[c_a, c_b])$, or $sc_x = f_x(d[c_i, c_j, c_k])$, etc.

In our example, we have a dataset-level summarization function f_g that collects information such as file type and filename. For feature-summarization we choose a single function f_b that abstracts the values in each column by their bounds, although other choices (such as a median or a Gaussian distribution) are possible. The resulting summary for each column contains its identity (i.e., name) and bounds, and may (as shown in Figure 2) contain other information, such as the data type of the contents and units. In our example we represent each individual column summary by a tuple: $\langle variable\ name, bounds, units \rangle$. Thus:

$$s(saturn01.ctd.201005) = F(saturn01.ctd.201005)$$

$$= \langle f_g(saturn01.ctd.201005), f_b(salinity), f_b(temperature), f_b(time) \rangle$$

$$sg = f_g(saturn01.ctd.201005)$$

$$= \langle \text{"saturn01.ctd.201005"}, \text{"verified"}, point(-123.8, 42.2) \rangle$$

$$sc_1(salinity) = f_b(salinity) = \langle \text{"salinity"}, [0..29.6], psu \rangle \ \ldots$$

Finally, we represent an archive of n datasets $D = \{d_1, d_2, \ldots, d_n\}$ by the resulting collection of summaries $S = \{s_1, s_2, \ldots, s_n\}$.

Query and Similarity. We now turn our attention to the representation of the query and the similarity function. A query is represented as a set Q of r query terms $Q = \{q_1, q_2, \ldots, q_r\}$. In our example, we represent each term q_i as a tuple of the form $\langle variable, range, units \rangle$. Our example scientist's query QE thus becomes:

$$QE = \{\langle \text{"temperature"}, [5\text{-}10], C \rangle,$$

$$\langle \text{"time"}, [2010\text{-}04\text{-}15..2010\text{-}07\text{-}15], date \rangle, \ \ldots \}$$

We use a similarity function *Sim_s* that, given a query *Q* and a summary *s*, produces a similarity score, $score_{Q,s} = Sim\_s(Q, s)$, representing the similarity between the query and the summary. We select similarity functions where the similarity computed over the summary is roughly equal to the similarity computed over the entire dataset. This restriction allows us to separate computing similarity into a one-time, offline function that summarizes the entire dataset, and a lightweight similarity function used during online search that operates only over the summary.

Further, we consider $Sim\_s(Q, s)$ to consist of the following steps: A function $Score\_s(Q, s)$ first calls a function $Match(Q, s)$ to match the query terms to features in the dataset summary. For each query term, *Match* identifies which feature in summary *s* should be compared to each query term; the feature returned often equates to a summarized column. *Match* returns a list of *r* pairs, each pair consisting of a query term and the corresponding feature from *s* (and, optionally, a weighting or confidence measure capturing the quality of the match between the query term and the selected feature). Here, for simplicity, we assume the feature is a single column, sc_i. *Score_s* selects and calls subsidiary scoring functions ($Score\_c_i$) for each query-term-and-feature pair returned from *Match*. An individual scoring function ($Score\_c_i$) operates on a single query-term-and-feature pair, and returns a score. *Score_s* composes the individual scores returned from these functions and returns an overall similarity score for query *Q* and summary *s*. Thus, *Sim_s* is the composition of a feature-matching function *Match* followed by a summary scoring function *Score_s*:

$$score_{Q,s} = Sim\_s(Q, s) = Score\_s(Match(Q,s)) \quad (5)$$

$$Match(Q,s) = Match(\{q_1,q_2,\ldots,q_r\}, \{s_g,sc_1,sc_2,\ldots,sc_m\}) \quad (6)$$
$$= \{(q_1, sc_i),(q_2, sc_j), \ldots (q_r, sc_k)\} \quad 1 < i, j, k < m$$

Here sc_i, sc_j, sc_k represent the individual features (columns) in the summary *s* selected by *Match*. If *Match* does not identify or choose a matching column, it may return no column for a query term. *Match* may also, as an extension, return more than one column for a single query term.

The purpose of separating *Match* from *Score_s* (and *Score_c*) is to separate different levels of abstraction and allow similarity to be assessed separately at each level. For example: we may have a second dataset, containing a variable "*temp*" but no variable "*temperature*". *Match* may have high confidence that in this case, "*temp*" is actually the same as "*temperature*" (perhaps based on matching units *C*), and therefore return it as the matching feature for the ""*temperature*" query term. For a third dataset, *Match* may make a different choice; for example if it (somehow) knew that "*temp*" in this case meant "*temperate*", it may return no feature for this query term. This feature-matching decision, and associated relevance or certainty, is at a separate level of abstraction from calculating the similarity of the contents of a temperature column to a query term for temperature, and concluding that, since the column contains values between the desired 5-10C, that the column contents have a high relevance score for this query term. Note that the behaviors of *Match* and *Score_s*, while partly independent, must be complementary.

In our example, *Match* very simply matches the variable name "temperature" in the query to the variable name "temperature" in our dataset summary, the "time" term in the query to the dataset's "time" variable, and the geospatial location portion of the query to the "location" information in the dataset summary.

On receiving the pairs of query terms and selected features from *Match*, *Score_s* applies a similarity scoring function *Score_c* component-wise to each pair. *Score_c* may be different for different columns or column types. For example, we currently use one scoring function for columns containing variable data, and a different scoring function that handles two-dimensional comparisons for geospatial data. Unit and datum transformations must also be performed at this time, or the scoring adjusted for non-transformed data as appropriate. In our example query with our example dataset, our *Score_c* function is called three times, as there are three query terms. We choose to compose the individual scores to give a final dataset score by averaging them.

Lastly, we return the result set containing the *k* highest-scoring datasets in our archive.

We believe the techniques in this work are not limited to any particular scoring or ranking function, as long as certain conditions are met (for example, that the scoring function is monotonic). These conditions will be described in future work.

3.2 Adaptability Across Multiple Scales

One of the issues for scientists in finding relevant data is the potential mismatch between the scales of the data they seek, the scales of observation, and the partitioning for convenient processing and storage of data. One example is the difference in scale between a single water sample, consisting of a single observation with a large collection of environmental variables from many tests run on the sample, as compared to a fixed observing station, with millions of observations each consisting of only a few environmental variables. Another example is a research cruise lasting for several weeks and covering hundreds of miles, collecting millions of observations. A scientist may be interested in only the observations collected while passing through her study area, a short segment of the entire cruise. A search performed only at the level of a summary of the entire cruise may cause a highly relevant subset of data to be thought not relevant on the basis of low similarity between the query and the entire dataset. In the case of the water sample, the individual observations may be stored in a larger dataset for storage convenience only, although the scientists logically regard each sample as separate; again, the overall similarity of the dataset to a single query may be low, masking the high relevance of a portion of its contents.

We wish to approximate a concept of "the most useful meaning-bearing unit" [30] for multiple constituencies of scientists, with each constituency having a different granularity in mind. For example, microbiologists tend to operate at different scales than oceanographers, but CMOP serves both communities. We want to mediate within our system between the convenient storage grouping from the perspective of the archive manager and the logically meaningful dataset for each scientist. To provide additional expressiveness across the range of possible datasets sizes and scales, we incorporate the idea of hierarchical metadata by allowing subset and superset relationships amongst entries within the collection of summaries. We allow a single dataset to be represented by multiple summaries, or a single summary to represent multiple (related) datasets. Each summary may correspond to the contents of a single physical dataset or file as stored, say, on a disk or in a relational database table; however, a summary may also represent a subset or superset of such a dataset. For example, a single table stored in a relational database may at times be treated as one or more datasets. This approach allows us to return the most relevant subset of the data in response to a scientist's query.

A set of summaries may be composed into a hierarchy by logically subdividing or composing summaries while retaining some knowledge of the relationship of their component parts. When a dataset is subdivided, we call the individual parts or subcompo-

nents *children*, and we call the dataset the *parent* of those children. Each child must be a strict subset of its parent, and we must be able to identify its parent (and vice versa). The children are not required to be disjoint, and the union of all children is not required to equal the parent. Each child must, however, itself fit the definition of a dataset. We call a summary with no parent dataset a *root*. Each tree consisting of a root, and, recursively, its children, we will for convenience call a *hierarchy*. The number of levels within the hierarchy is not limited, nor need it be equal on all paths. The segmentation of parents into children is neither limited nor prescribed, and is left as a design choice to each application or dataset collection within an application (as discussed in our description of our prototype in Section 2). A *metadata catalog* is a forest of hierarchies representing an archive.

3.3 Approach

Given the model described, we can now explore the feasibility and utility of the model in supporting search over archives of scientific data. Can scientists express their information needs as a set of query conditions, as described? Can we create a catalog of dataset summaries from an existing archive? Is it possible to identify a similarity measure that will resonate with scientists, and apply the similarity measure to the dataset summaries?

Cognitive science [20] has long recognized that people frequently use distance as a metaphor for similarity. Tversky and Gati [33] point out that "the notion of similarity – that appears under such different names as proximity, resemblance, communality, representativeness, and psychological distance – is fundamental to theories of perception, learning, and judgment". They note that similarity relations have been dominated by geometric models; these models represent points in a coordinate space, generally assumed to be Euclidean, and the distance between the points is taken as a measure of their similarity. In the field of spatial cognition, Fabrikant [10] notes that a spatial "near-far" image schemata is often used as an abstraction for similarity (for example in estimating distance between documents based on content similarity), and that this metaphor carries over into multiple (non-spatial) dimensions. In further support, Lakoff notes that interpreting time as distance is a common metaphor [20]. While it is well known that people are inaccurate in their estimates of absolute distance, research shows that they are relatively consistent in ordinal rankings [25]. Thus, a measure that adequately replicates the ordinal rankings given by users would suffice for our system. We summarize results of our research into these questions in Section 4.

If we can achieve these things, we are left with still another challenge: is it possible to provide interactive response times, as we have come to expect from document search, over a metadata collection representing a scientific data archive of current sizes? We describe our thoughts on this question in Section 5.

4. USER EXPERIENCES

We performed two user studies, which we briefly review here. Our initial user study was intended to test several hypotheses, key amongst them being whether our intended users could relate to a brief summary of a dataset, such as the time and geospatial bounds of contained data. Further, if we asked our intended users to identify which of several datasets was the most similar to their query, would a population of users make similar choices even with several query terms? This study was performed using a paper survey instrument, in order to remove tool effects from the responses. Two separate user populations were studied; twenty scientists from CMOP, and twenty Information Technology professionals and students. We found that the "similarity as distance" metaphor [33] was applied quite consistently by our respondents across the spatial and temporal dimensions tested; this is consistent with findings in the field of spatial cognition [10]. Further, the results for these two populations were almost identical.

The second user study was intended to assess the quality of our similarity measure, including non-spatiotemporal variables, and to test the utility of the tool. This study consisted of 12 scientists drawn from the pool of intended users. Utility was measured using a qualitative assessment of user satisfaction on a 7-point Likert scale, using questions adapted from Su [31]. Respondents were asked to issue queries with different characteristics (space or time only; space, time and a variable; space, time and a variable with a specific range and units), drawn from their own research and information needs. Each query type was separately assessed. Respondents reported statistically meaningful high satisfaction with the query tool (median response of 6.5, interquartile range 1.1) and the results presented. Respondents reported that they could not have found the data as quickly via other means available to them (median 6.5, interquartile range 0.5).

The question with the highest variance (median response 6.5, interquartile range 2.5) addressed respondent confidence in the completeness of the tool's result sets. Based on examples brought to us by respondents, we believe the high variance comes from several factors. In some cases known to us, the sought-for data had not yet been processed by the metadata extractor, and thus was not in fact available for search. In some additional cases, the respondent had in mind a dataset that did not, in fact, exist; for example, when a respondent had mistaken the variables, contents, location, or timing of certain sought-for data

The second study also asked respondents to assess relevance of a randomized subset of returned results. Precision at rank 10 gives a measure of the number of relevant documents found in the top ten returned [35]; our mean precision at rank 10 was 0.96±0.19. Mean reciprocal rank (MRR) measures the average position of the first relevant document found [35]; overall MRR was 0.95.

We found our results and the enthusiasm of our scientists encouraging. Several of our respondents bookmarked the user study code and continued to use it for searches until the recent production launch of the tool, to find data that would be lost when using the more rigid interfaces previously available to them.

5. PERFORMANCE AND SCALABILITY

We are now exploring the performance and scalability characteristics of our prototype. This section describes the results of initial experiments and our current plan for this portion of our research.

5.1 Initial Performance Experiments

While performance of Data Near Here is sufficient for the current CMOP archive (most queries take a few seconds; a billion observations are currently cataloged), we are interested in the effects of further scaling. Scaling may occur as a result of an increasing number of queries, of increasing complexity of the queries issued, or with an increase in the number of metadata records. Growth in the metadata records may come from a number of sources: growth in the underlying observational data occurs daily. Adding metadata records for simulation output or other classes of data will increase the size of the metadata catalog by some factor. Even excluding organic growth, however, changes in hierarchical structure may have a large impact: a recently-discussed move from monthly to daily datasets for fixed stations would increase these catalog records by a factor of 30. Addition of metadata for other archives will also increase the overall metadata count. We are curious about the effect of growth on interactive response times,

and about what techniques could compensate for any slowing.

In addition to performance effects caused by the growth in catalog entries, the characteristics of the catalog entries themselves are likely to change user response times. In particular: organizing the datasets in a hierarchy provides users with data in scales matching their research interests. It can also improve (or detract from) performance. How do different hierarchy patterns (for example, deep versus shallow hierarchies) affect response?

Another major performance aspect is the effect of varying the queries themselves. We do not yet know what the "usual" number of query terms will be; but we know that this number will have a large effect on response times. We speculate that a common number may be around 5, based on user studies in Internet search [1, 4, 19]. Many query systems, including ours, suffer from "the curse of dimensionality" [17]. Similarity measures that are based on distance, such as ours, are affected by the dimensionality of the data space and by variations in the data distribution; these variations affect the quality of the results and the performance characteristics. How does increasing the number of query terms affect response times in our system?

We performed initial performance analysis using a set of arbitrarily selected test queries, prior to implementing any methods to improve performance. We analyzed the proportion of elapsed time the system spent in the query process steps as shown in the sequence diagram in Figure 4, with the following conclusions:

Issue Search Request (steps 1 and 2): The user interface time spent in preparing the search request (step 1) and in sending the request to the back end (step 2) is negligible.

Search and Rank Results (step 3): The search-and-rank task contributes the greatest proportion of the response time and shows the greatest variability. This time is made up of SQL query time (multiple queries may be performed) plus a (relatively) fixed time per result returned from any SQL query. As the size of the database and the complexity of the queries grow, this time will further dominate the overall response time.

Send Results (step 4): The network delay is driven by certain factors outside our control (distance, connectivity, bandwidth), and by the amount of data transmitted between the search engine and the user interface. We could possibly reduce the data transmitted by compression; however, it is unlikely that such a change would make a dramatic difference to the overall response time.

Build Results Page and Map (step 5): The time spent in step 5 is primarily driven by the number of elements we draw on the map. We have now artificially limited this time by defaulting the items we draw to the top 25 results returned, as a good compromise between providing sufficient mapped results and taking too long. In the absence of mapping delay, the primary driver is the number of items returned from the search. We provide default "minimum and maximum items returned" numbers to the back-end. The user can request additional items using a "get more" button, if she wishes. Alternate user interface designs are possible that would allow data to be displayed with less delay. These designs are not the subject of our research.

The performance of the user interface and network delay will not be affected by substantial growth in the underlying metadata collection. However, the performance of the search engine is directly affected by the size of the metadata catalog and by the techniques used to identify the top-ranking datasets. Therefore, in order for system utility to continue despite substantial growth, search engine performance must be characterized and understood. Thus, the search engine component is the primary focus of our performance and scalability research.

As currently implemented, the search engine uses several techniques to achieve the current interactive response times over the current metadata catalog. They are:

- A filter, implemented in SQL, limits the records returned from the database to those ranked above an estimated similarity cut-off score.
- Naïve relaxation of the similarity score: We begin with a cut-off score of 95. If the current similarity cut-off score does not produce a specified minimum number of results, we drop the cut-off score by 10 and re-issue the query.
- An approximation technique is used to simplify and limit the number of distance calculations performed.

We know that as terms are added to the query, response times increase, and that incorporating a cut-off improves response times on the more expensive searches [22]. An area for further work is determining how to initially set the cut-off threshold for a given query; as Chaudhuri et al. [7] note, there is as yet no satisfactory answer to estimating result cardinalities and value distributions, leaving this as an open area for research.

Response times are affected by the physical design. We use a simple structure consisting of one table with the dataset-level information and another with information about the variables; this approach allows new variable names to be added easily into the database, as they are easily appended to the variables table (unlike adding new columns). On the other hand, search performance is limited by this design due to the use of joins and pivot tables to match dataset-level and variable-level query terms.

5.2 Scaling the Data

In all cases, our goal is to maintain query response in the current "interactive response" range, while the size or complexity of the underlying metadata collections varies. Of the various factors known to affect performance, we have selected three for further exploration: physical data design; query evaluation techniques; and the hierarchy.

Physical Data Design. Our initial data design was chosen for ease of implementation, but appears to constrain our performance. We plan to test several alternate physical data designs.

Query Evaluation Techniques. We will further explore the use of top-K query evaluation techniques in our setting. Ilyas et al. [18] classify over 20 top-K approaches in a taxonomy based on their capabilities and assumptions about the environment in which they

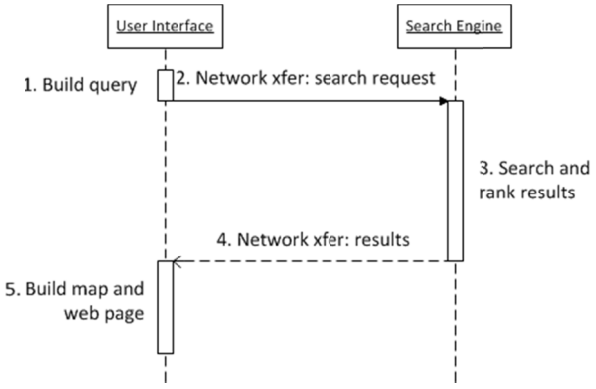

Figure 4. Search Process: Sequence Diagram

operate. Six techniques identified in their paper match the characteristics of our design; that is, the subset of techniques classified as: top-k join query (results of a join with the largest combined scores); exact methods over certain data (uncertainty is not required); both sorted and random access are available; we desire an application-level technique (as opposed to those that modify the query engine); and monotone ranking functions. Additional literature review may add to the techniques to be explored.

The Hierarchy. We will analyze the effect of different "shapes" of the hierarchy over different metadata collection sizes, to identify where the hierarchy helps or hurts performance. For example, we will compare single-level metadata with deep hierarchies.

We have also made the following decisions. We will concentrate on memory-resident metadata catalog sizes. We believe this choice is reasonable since during search our metadata is read-only, and the index can be split across multiple servers. Further, the metadata is, in itself, an index to the data; access to the data itself is initiated by the scientist when she selects a download or visualization link. We will use queries "contributed" by our users; these are captured (without identifying characteristics) in a query log. We will use real (possibly replicated) data from our archive. The content of the current metadata catalog is highly skewed geospatially, in time, and in terms of the variables contained. We expect other archives to have the same skewed data pattern. Even where the time and geospatial distribution is consistent, for example, the values of environmental variables (such as temperature or salinity) may be skewed.

5.3 Discussion

The negative impact of increasing system response time on users was recently reaffirmed by Schurman and Brutlag [29]; adding server delays to existing server response times decreased user search activity, satisfaction and, in their case, revenue. Page size returned was shown to not affect these measures. These findings support our focus on understanding the latency effects of changes in collection size and query complexity.

We wish to understand the performance of a single server; if single-server performance is sufficient for the expected workload, no additional scaling approaches are needed. When single-server performance is found insufficient even after optimization, throughput can be further scaled by adding servers; we discuss several of these approaches here. The scaling design must consider how the workload is to be distributed across these servers. Tomasic and Garcia-Molina [32] summarize the two basic strategies for distributing an inverted index over a collection of query servers: so-called *local inverted files*, where each server is responsible for a disjoint partition of the documents in a collection, and so-called *global inverted files*, where each server is responsible for a disjoint partition of the terms (but across all documents). They simulate performance of different configurations of global and local indexes and conclude that local inverted files provide the best performance. The performance of the servers containing the disjoint index partitions therefore limits overall system performance. We expect to see the same.

Building on their work, Cacheda et al.[5] and Long and Suel [21] describe the use and performance of "query integrators" to distribute queries to and combine results from multiple servers with local inverted files. Cacheda et al.[5] compare the performance of searching a terabyte of text on three different multi-server architectures: distributed, replicated and clustered. Depending on the query characteristics, either the replicated or the clustered configurations performed better. Latency is increased by the need to route the queries or aggregate the results (depending on the approach taken), while throughput of the entire system, that is, the number of queries that can be handled simultaneously, is increased. While we are not using inverted files to index our data, the scaling approaches otherwise apply. We would see the same effect of increasing response time as compared to a single system.

The recent move to cloud-computing infrastructures has led to new approaches that perform well at larger scales. For example, Wang et al. [36] describe a multi-dimensional indexing scheme, RT-CAN that efficiently supports multi-dimensional query processing in a cloud. RT-CAN creates a distributed global index that is used to identify the local nodes containing relevant data. The distribution approach may be based on geographic location or on data ranges. They demonstrate their system using point, range and nearest-neighbor queries, using a 128-node network. We believe this scaling approach applies to our work.

6. RELATED WORK

There is much research [9, 23] into ranked relevance of unstructured text documents and XML with text queries; in contrast, our work focuses on numeric data ranges. Numbers in HTML tables can be extracted and searched [34], but this work focuses on extracting additional semantics; numbers are also searched for by Agrawal and Srikant [2], but both these approaches assume each "document" is small by our standards. To our knowledge, ours is the first application of these ideas to collections of large, heterogeneous datasets.

Scientific archives support searching for text in metadata associated with datasets [26, 27]; these searches are primarily Boolean in nature. Most geographic search systems [14, 15] score items based on word matches against metadata without considering the temporal span or geographic content of the items returned. State-of-the-art geospatial portals [12, 13] allow searches using both geographic and temporal criteria; generally, three spatial tests are supported (the map view *intersects*, *mostly contains*, or *completely contains* the dataset), and temporal searches appear to be Boolean *contains* tests. In contrast [24], we explicitly rank returned items based on combined temporal, geographic and variable "distance" of the dataset contents from the query.

The work perhaps most similar to ours in scope and spirit is by D'Ulizia et al [8]. Like us, they are interested in providing results ranked in similarity across a combination of geospatial and non-geospatial attributes. Each attribute is assumed to have a single value, and the attribute and values are matched via known ontologies and taxonomies. They provide alternate queries (rather than results) by relaxing the query terms if no matching terms are found, and they compute the similarity of a "relaxed" (approximate) query to the original query (rather than to the database contents or query results). All their data exists within their database, whereas we work with summaries of data.

The spatial component of our work draws on research in the field of spatialization of data [10, 30]; despite the name, the data researched is not spatial in nature, but is nevertheless represented as points in a vector space. Spatial cognition researchers have shown that judging relative distance between individual spatialized data points is practical, and that similarity represented as relative distance is naturally understood. We apply their notions of distance more broadly to large sets of combined temporal, spatial and environmental-variable values. Although their work has identified anomalies and inaccuracies in user perceptions at the detail level, we believe a fast, simple approximation has significant value.

Su, Al-Maskari [3, 31] and others have discussed the relationships between user satisfaction and IR measures. Our user studies were informed by their work, and we adapt their methods to dataset relevance evaluation and validating the utility of our prototype.

7. CONCLUSION

We are encouraged by the enthusiasm of the scientists for our work, and by their requests for additional data to be searchable through our tool. We believe our approach to searching scientific data is broadly applicable. As data volumes and heterogeneity grow, need for tools such as "Data Near Here" will only increase.

A data archive's value can be measured by the use and reuse made of its contents. If relevant data cannot be found in the archive even when it is stored there – if the data is "lost" – the archive's value is diminished. By adapting techniques first developed for similar challenges in the world of text documents, we believe that – at least for scientific data – what was lost, can still be found.

8. ACKNOWLEDGMENTS

This work is supported by NSF award OCE-0424602. We thank CMOP staff and scientists for their support.

9. REFERENCES

[1] Ageev, M. et al. 2011. Find it if you can: A game for modeling different types of web search success using interaction data. *Proceedings of SIGIR* (2011).

[2] Agrawal, R. and Srikant, R. 2003. Searching with numbers. *Knowledge and Data Engineering, IEEE Transactions on*. 15, 4 (Aug. 2003), 855 – 870.

[3] Al-Maskari, A. et al. 2007. The relationship between IR effectiveness measures and user satisfaction. *Proc. of SIGIR* (2007), 773–774.

[4] Aula, A. et al. 2010. How does search behavior change as search becomes more difficult? *Proc. of the 28th International Conference on Human Factors in Computing Systems* (2010), 35–44.

[5] Cacheda, F. et al. 2005. A case study of distributed information retrieval architectures to index one terabyte of text. *Information Processing & Management*. 41, 5 (2005).

[6] Center for Coastal Margin Observation & Prediction (CMOP): *http://www.stccmop.org/*. Accessed: 2011-04-17.

[7] Chaudhuri, S. et al. 2005. Integrating DB and IR technologies. *CIDR'05*. (2005), 1–12.

[8] D'Ulizia, A. et al. 2009. Approximating Geographical Queries. *Journal of Computer Science and Technology*. 24, 6 (2009), 1109–1124.

[9] Demartini, G. et al. 2010. Overview of the INEX 2009 entity ranking track. *Focused Retrieval and Evaluation*. (2010).

[10] Fabrikant, S.I. et al. 2004. The distance-similarity metaphor in network-display spatializations. *Cartography and Geographic Information Science*. 31, 4 (2004), 237–252.

[11] Gartner Says Solving "Big Data" Challenge Involves More Than Just Managing Volumes of Data: 2011. *http://www.gartner.com/it/page.jsp?id=1731916*. Accessed: 2012-06-28.

[12] Geospatial One Stop (GOS): *http://gos2.geodata.gov/wps/portal/gos*. Accessed: 2011-01-19.

[13] Global Change Master Directory Web Site: *http://gcmd.nasa.gov/*. Accessed: 2011-01-19.

[14] Goodchild, M.F. and Zhou, J. 2003. Finding geographic information: Collection-level metadata. *GeoInformatica*. 7, 2 (2003), 95–112.

[15] Grossner, K.E. et al. 2008. Defining a digital earth system. *Transactions in GIS*. 12, 1 (2008), 145–160.

[16] Hey, T. and Trefethen, A.E. 2003. The Data Deluge: An e-Science Perspective. *Grid Computing: Making the Global Infrastructure a Reality (eds F. Berman, G. Fox and T. Hey)*. John Wiley & Sons, Ltd, Chichester, UK. 809–824.

[17] Houle, M. et al. 2010. Can Shared-Neighbor Distances Defeat the Curse of Dimensionality? *Scientific and Statistical Database Management* (2010), 482–500.

[18] Ilyas, I.F. et al. 2008. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys (CSUR)*. 40, 4 (2008), 11.

[19] Jansen, B.J. et al. 2000. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing & Management*. 36, 2 (2000), 207–227.

[20] Lakoff, G. 2000. *Where Mathematics Comes From*. Basic Books.

[21] Long, X. and Suel, T. 2003. Optimized query execution in large search engines with global page ordering. *Proc. of the 29th VLDB Conference* (2003), 129–140.

[22] Maier, D. et al. 2012. Navigating Oceans of Data. *Scientific and Statistical Database Management* (2012), 1–19.

[23] Manning, C.D. et al. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[24] Megler, V.M. and Maier, D. 2011. Finding Haystacks with Needles. *Scientific and Statistical Database Management* (2011), 55–72.

[25] Montello, D.R. 1991. The measurement of cognitive distance: Methods and construct validity. *Journal of Environmental Psychology*. 11, 2 (1991), 101–122.

[26] Pallickara, S.L. et al. 2010. Efficient metadata generation to enable interactive data discovery over large-scale scientific data collections. *2nd IEEE International Conference on Cloud Computing Technology and Science* (2010), 573–580.

[27] Rajasekar, A. and Moore, R. 2010. Data and metadata collections for scientific applications. *High-Performance Computing and Networking* (2010), 72–80.

[28] Salton, G. 1968. *Automatic Information Organization and Retrieval*. (1968).

[29] Schurman, E. and Brutlag, J. 2009. The user and business impact of server delays, additional bytes, and HTTP chunking in web search. *Proc. Velocity: Web Performance and Operations Conf.* (2009).

[30] Skupin, A. and Buttenfield, B.P. 1996. Spatial metaphors for visualizing very large data archives. *Proceedings of GIS/LIS '96* (1996), 607–617.

[31] Su, L.T. 1994. The relevance of recall and precision in user evaluation. *Journal of the American Society for Information Science*. 45, 3 (1994), 207–217.

[32] Tomasic, A. and Garcia-Molina, H. 1993. Performance of inverted indices in shared-nothing distributed text document information retrieval systems. *Proceedings of the Second International Conference on Parallel and Distributed Information Systems* (1993), 8–17.

[33] Tversky, A. and Gati, I. 1978. Studies of similarity. *Cognition and Categorization*. 1, (1978), 79–98.

[34] Venetis, P. et al. 2011. Recovering semantics of tables on the web. *Proc. of VLDB 37*. 4, 9 (2011), 528–538.

[35] Voorhees, E. and Tice, D.M. 1999. The TREC-8 question answering track evaluation. *Text Retrieval Conference TREC* (1999).

[36] Wang, J. et al. 2010. Indexing multi-dimensional data in a cloud system. *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data* (SIGMOD '10) , 591–602.