A Distributed, Graphical, Topic-oriented Document Search System

John Light
Intel Architecture Labs, Hillsboro, Oregon
Intel Corporation
jjlight@ibeam.intel.com

Abstract

With the increasing numbers of documents available from the Internet and other sources, finding anything is becoming increasingly difficult. This paper presents a new approach to finding documents that relies on compelling graphic presentations to the user. By creating compact representations of documents that can be easily transferred between computers, the approach allows the search process to occur on the user's desktop, distant from where the original documents are stored or where the indexes are created. Besides allowing searching to scale better to large numbers of information consumers, this allows the full graphical capabilities of the desktop computer to be applied to the problem. The ability to search for specific user keywords is replaced by user selection of topics from a list created by expert topic designers. The approach complements keyword searching as a way of finding information.

Keywords: information retrieval, information visualization, Internet, document search, personal computer

1. RECOGNIZING THE PROBLEM

Most readers have had the opportunity to find something on the World Wide Web (WWW). Some readers have used research services such as Nexis® or Dialog® to find information in a document repository or journal back-issues. Using current technology to find interesting documents in an ocean of available documents is a hit-or-miss proposition: sometimes it works, and sometimes it doesn't. When it works, the feeling of power (or magic) can be exhilarating. When it doesn't, frustration is accompanied with the thought that if you had only been a little smarter, it would have worked.

No single solution is going to overcome the inherent difficulty of finding document needles in today's cyberspace haystacks. The magnitude and complexity of the problem is just too large (and growing too fast) to expect to find a single solution to this important problem. Not only are document repositories growing with each passing day or month, the number of repositories is growing in the same time scales. The result is exponential growth of information.

The scale of the information overload blinds us to the real problems. I will make a stab at describing the real problems without pretending to fully enumerate or understand them.

- Along with total information growth there is an equivalent growth in specialized areas of discourse. There are not just more scientific journals, there are more scientific disciplines, each with their own terminology and ways of talking about issues. While there is often significant overlap in vocabulary, it is common for two disciplines to use different words to describe the same thing, and nearly as common for a term to refer to different things in different disciplines. Sometimes the ambiguity is intentional (e.g., by using another discipline's terminology as metaphor), and sometimes the ambiguity is unintentional (e.g., resulting from subtle shifts in related discourses).
- No one can keep up with all the words and phrases in all the different forms of discourse taking place today. While it might be possible for someone with eidetic memory to memorize a dictionary, the universe of discourse is hugely larger than even the Oxford English Dictionary because of the specialized use of phrases (combinations of words) in nearly every form of discourse. In effect each conversation creates new *words*, consisting of strings of old words, on a daily, weekly and monthly basis.
- Some of the most interesting and important searching being done today is across disciplines. Whether it is done by someone who is a novice or expert in his own discipline, these searches are in a space where the searcher doesn't really know or understand the vocabulary. Historically, some of our greatest inventions have resulted from connecting disparate disciplines, so supporting searches in foreign domains is critically important. Our current search methods, which rely heavily on the user's ability to pick individual words, make that hard.
- Search is currently an entirely mechanical process. Current search mechanisms rely on the creation of

massive indexes on large computers. (A document index can be 50-300% of the size of the original document set.[3]) The actual search process accesses the index using clever algorithms that don't understand the underlying content as anything more than a list of words. When the process fails, most searchers have no recourse. Those who subscribe to services such as Lexis-Nexis can fall back to the support of experts. These experts typically support one or a few document sets with which they are intimately familiar. For a price, these experts perform research or suggest search terms and strategies. For most searches, for most people, no expert help is available.

- Search performance doesn't scale well. Because the indexes are so large, they must be kept near (in a bandwidth sense) the indexing computer. Because the search process is inherently processor intensive, episodic, and intimately connected to the index, searching is typically performed on servers which support multiple users. For small numbers of users, search performance will be highly variable, depending on how many other search requests are queued or in process. For large numbers of users, the law of large numbers reduces variability, but providing large enough computers to reliably keep request queue lengths short is a very expensive proposition.
- Searching currently doesn't exploit the large amount of processing power on our desktops. Most current desktop computers are as powerful as specialized search machines of ten years ago, but we aren't using them to help with the searches. In fact, most searches today use our desktop computers as nothing more than dumb terminals, supporting typing of search requests and display of search results. Ironically, some desktop machines that act as dumb terminals are actually more capable than the central server doing the search. This is a tragic waste of resources, mostly of the human resources waiting for the search results.
- Searching doesn't effectively use peoples' skills. While it is well known that humans are primarily visual animals, our most effective visual skills involve recognition of sizes, shading, colors and movement. Our ability to read is only a side benefit of our hundreds of thousands of years of evolution, and carries with it a high cognitive load. Current search techniques communicate with us entirely with text, guaranteeing slow communication speeds both in and out.
- Searching has no memory. Each time you perform a textual search, you start over from scratch. While some search environments provide tools to keep you from having to type words over again, the actual search process typically starts over from the beginning every time you make a search request. To illustrate the problem, consider this example. You search for "banana" on AltaVista and get 70000 matches. You search for "pear" and get 20000 matches. What you want to do is search the union of the results of those searches for the word "compote". Unfortunately, you can't really do that easily. Some search engines (AltaVista is one) allow you to construct syntax that will allow you to find the 500 entries that qualify. Others don't. In either case, few people know how to construct the advanced query, and the advanced query starts from scratch as if the first two queries had never been done.
- Many people don't have effective access to some of the documents they need. Many document repositories contain valuable intellectual property. The owners of Lexis-Nexis pay for the libraries they maintain and expect to be compensated for the use of their contents. Many of us who don't have subscriptions to Lexis-Nexis have wondered whether the information we need is in one of their libraries. Since the only way to "browse" their libraries uses their search engines, casual use is quite expensive and in most cases prohibitive. Yet if I knew they had a document of interest to me, I might be willing to pay for it. This state of affairs has two unfortunate consequences. First, I don't have access to documents I might need. Second, Lexis-Nexis doesn't sell as many documents as they could.

The currently most common method of searching document repositories, indexed word searching or keyword searching, is a powerful tool when used by a knowledgeable individual in the pursuit of an appropriate goal. But it has serious drawbacks under many circumstances, and our continuing reliance on it will limit our ability to take enjoy the fruits of the information revolution.

2. QUESTIONING ASSUMPTIONS

While some work has been done on solving nearly every one of the problems noted in the previous section, each has been approached as a point improvement to the current regime, with little or no view to new approaches to the problem. Most of the search engines today look alike from a user's point of view. They all take about the same form of input, and they all present the results as a short list of the most relevant titles out of a large total. Each has some innovations that improve on the common denominator, but the differences are minimal for most users.

The characteristics of the dominant paradigm make major incremental improvements unlikely.

- Each search engine must be able to be totally general. That is, it must include positional keyword search, or it will be judged inferior. This places a large minimum processing requirement on the engine.
- Text indexes can be distributed, but they are inherently very large and must be kept intact. This limits distribution to well connected other servers, with no way to do any significant work on a desktop.
- Text retrieval is currently very Aristotelian. That is, answers are judged as either right or wrong. It is very difficult to generalize Aristotelian logic to perform loosely defined tasks.
- Specifically, the Aristotelian nature of query results makes any sort of graphic presentation very difficult.
- Since the search process takes place on a shared server, the ability to customize the search process or maintain search state for individual users will always be minimal.
- Users and their needs are seen as identical or at least qualitatively similar. The idea that searching might use multiple mechanisms to meet varying needs would

require recognition of the failures of keyword searching.

I believe that it will be difficult to move to new search mechanisms without reconsidering many of our assumptions at the same time. Trying to reconsider assumptions one at a time doesn't provide enough degrees of freedom to break out of our current stagnant state.

In my work on search, I made the following group of assumptions that run counter to current practice.
The union of these counter-assumptions is what makes the invention work.

- For many searches, especially early in the discovery process or done casually, the complete generality of keyword searching is neither necessary nor desirable.
- Searchers can deal with large results sets (>1000) if the results are presented graphically, with cues to allow the visual cortex to make discriminations.
- Early bound quantifications of relevance can be very useful if they are applied to a sufficiently narrow subject area.
- The problem of information overload is so important that quantifications of relevance need only be provably useful, not provably correct.
- Incorporating expert knowledge in the search process will make searching more effective.
- Modern desktop computers have adequate functionality to perform a portion of the search process.
- The logistical problems of moving much of the search process to the desktop will be overwhelmed by the advantages.
- By involving the searcher in the search process rather than just presenting search results, the process is made more effective.

The one assumption I am not questioning is the value of full-text indexed searching. There will always be circumstances in which being able to search for arbitrary combinations of words will always be needed and the best available tool. I am looking for mechanisms that complement keyword searching, not replace it. Right now it's the only game in town.

I want to acknowledge some more localized innovations in document processing and visualization which have helped pave the way for acceptance of new approaches to searching.[1 2 4 6 7]

3. USING THE TECHNOLOGY

The system I am proposing is only possible because of recent advances in technology. Below I enumerate the technological changes that have made the system possible, many of which have happened only recently.

- Desktop processors of sufficient processing power (> 100 MIPS).
- Greatly reduced storage costs, allowing significant desktop stores (> 1 gigabytes).
- Affordable 3D acceleration hardware.
- Ubiquitous networking with adequate bandwidth (> 100 kilobytes/second).
- A general medium for publishing data sources (the World Wide Web).

We need to recognize that the landscape has changed and use the new resources to solve our continuing and growing problems.

4. INVOLVING THE USER

The most unexploited resource is the one that has been available the longest. People have an uncanny ability to make discernments that are beyond the reach of the most powerful computers. Current technology doesn't use that ability. The user is seen only as the consumer of the technology, providing complete queries and receiving the mechanical judgment of the search engines. The user is kept at arm's length, unable to see or participate in the process of finding documents. If the user doesn't like what he gets, he can only try again, formulating a new incantation which he can hope works better. Keeping all searching on servers necessitates this reality.

I propose that more of the search process be moved to the desktop. Not only does this make better use of an underutilized computing resource, it has the advantages that it allows graphical presentation of search results and allows the user and his powerful brain to become an integral part of the search process.

The primary difficulties with involving the user are the disparities of bandwidth in the delivery system. The reason things are the way they are is that search is a data-intensive task, and until recently getting large amounts of data to a user's desktop was a difficult task. The current keyword search paradigm minimizes bandwidth between the server and the desktop, at the cost of flexibility. Since it is difficult for the user to absorb more than ten to a hundred textual query responses, the need to make sure they are the best responses is paramount, reinforcing the need for full-text indexed searching. I propose to break this system level feedback loop.

The primary requirement for desktop searching is a means of representing document set contents that is compact enough to be transported to the desktop and stored there. In general, it will not be possible to move full-text indexes to the desktop, for both transport and storage reasons. If they could be moved there, modern desktop computers are powerful enough to use them, but the costs of transport and storage would typically overwhelm the value they might provide.

To make moving a document set representation to the desktop, I propose the following mechanism:

- Recognize the underlying dimensionality (N) of the document set.
- Represent each document of a set as a vector in the underlying N-dimensional space.
- Each element in a document vector is a scalar that purports to show the strength of that document in that dimension.
- Apply a minimum threshold to each dimension so that most scalars are zero for most documents.

- Include an identifier that allows access back to each original document.
- Include commonly accessed information about each document to minimize references back to the original document set (e.g., title, size, date, etc.).

This representation of a document set, which I call a *skeleton*, has the following functional characteristics:

- Carefully encoded, it will be about one percent of the size of the original document set.
- Using non-Aristotelian logic, the preponderance of zero vector entries allows powerful filtering operations.
- The non-zero vector entries can be used in a variety of ways to characterize and visualize the document set as a whole as well as the individual documents.
- The additional information allows useful handling of the documents without having to access the original document set, which may be remote, unavailable, or expensive to use.
- The sparseness of most document vectors allows a highly compact vector representation.
- The computing cost of generating the *skeleton* is linear with the size of the document set. This allows handling of huge document sets cost effectively on a regular basis. (Most current mechanisms are M log M, and some are even M^2, where M is the number of documents in the set.)
- Most processing operations on the *skeleton* are also linear with size, allowing effective handling of large document sets on the desktop. Interesting non-linear operations can often be performed after filtering the documents to reduce the working set size.

The second requirement for desktop searching is an effective graphical visualization environment. This area is largely unexplored for document sets, because until now there were few means of attaching scalar significance to bags of text. This paper concentrates on building the data infrastructure needed by the system, so my associated work on visualization is not detailed.

As an example of the available visualizations, I chose to plot documents in any three of the document set dimensions. This Cartesian graph of documents has the advantage that it is simple to implement in a variety of 3D graphics environments, and its form is instantly recognizable to most people who took high school algebra. I first implemented this graph in VRML and have since created a more robust visualization using OpenGL.

The third requirement for desktop searching is a process that maintains and manipulates document set state while interacting with the user. For example, once the user has examined a document, it shouldn't be presented as a new "find" each time the user makes another query. For another example, the user should be able to identify subsets of the total document space that are of interest for further search. Neither of these capabilities is commonly provided or easy to provide in server based search systems.

5. EXPLOITING EXPERTS

A bullet in the preceding section blithely refers to "a scalar that purports to show the strength of that document". Another bullet requires that we "recognize the underlying dimensionality of the document set".
Both of these concepts are fundamental to the successful operation of the proposed system, but neither is well understood.

Both of the concepts are the subject of intense and continuing research in the Information Retrieval (IR) community. Most of their work has concentrated on mechanisms by which computers can *discover* dimensionality and *measure* the dimensions of a document with respect to other documents. This is necessary because their goal is to provide completely automated systems that are provably *correct.*

I propose a largely automated system that uses expert information that is provably and intentionally *subjective.* The application of a human viewpoint is an additional advantage to the system, not a drawback. One way to look at the expert contribution is as that of an *editor* of a publication. In effect, the *skeleton* is a specific editor's view of what is important and interesting in a document set. I believe this is appropriate for searching subjects that are not within a user's field of expertise, which include most subjects for most users. Again, users who find this restrictive can use full-text indexed search.

The primary contribution of the expert is to identify a large list of narrow topics within the document set. If the topics are narrow enough, they can be combined in various ways to construct complex queries into the document set. Ideally, the scalar strength of each topic will be independent of the scalar strength of other topics, making the dimensions orthogonal. In practice, this is not always possible, but it isn't hard in most cases.

The second, and more time-consuming, contribution of the expert is a description of the vocabulary used to discuss each topic. I have developed a vocabulary-description language that allows concise descriptions of topic vocabulary. A topic is described primarily by a list of words and phrases that are specific to the topic. The careful use of phrases allows minimization of false positives. The skill and knowledge of the expert minimize false negatives. The topic description is used in a formula from the IR category *similarity functions* to generate the scalar topic evaluation.

The third contribution of the expert is to establish a threshold for each topic. Two issues determine the threshold. First is the tradeoff between false positives and false negatives. Second is the tradeoff between positives and *skeleton* size (the more zero vector entries, the smaller the *skeleton*). In practice, selection of a threshold is not difficult because the design of the vocabulary description language minimizes false positives and *skeleton* size.

Each time a new document set is targeted by the system, topic profiles (dimensions) must be created for it. The collection of topic profiles for a document set is called *reduction rules*, after the data reduction step of traditional scientific data visualization. The *rules* reflect the expert's knowledge about the targeted document set. They don't have to be constructed from scratch every time. To the extent that a document set is similar to a previously profiled document set, the *rules* can be carried over. As an expert develops *rules* for various document sets, the work needed to build *rules* for new documents sets decreases.

6. SUMMARIZING DOCUMENTS

After the *reduction rules* are written, all further processing is automated. That is, no further intervention by the expert is needed unless a change in the document set necessitates modifications to the *rules*. The processing steps are similar to the steps used in traditional IR.

1. Recognize the terms in the document.
2. Recognize specified phrases.
3. Remove unwanted terms.
4. Replace most terms with their stems.
5. Compute a similarity function with each topic profile to create the topic scalars.
6. Apply thresholds to each topic scalar.
7. Construct a compact topic vector from all the above-threshold topic scalars.
8. Attach the topic vector to the common document information to form a document *surrogate*.
9. Collect all the document *surrogates* into a *skeleton*.

Step 2 is not currently done in existing IR systems. The recognition of phrases, when it is done, is performed during the use of the index. Steps 5 and 6 involve early binding of a partial query (the topic profile) and is not done in current IR systems. Steps 7, 8, 9 replace the index inversion step of most IR systems and is not done currently. Note that none of the steps is particularly compute intensive or dependent on the contents of other documents, allowing efficient processing of very large document sets.

The choice of how much common document information to include depends on the user's ability to access the main document set. If it is readily available at high speed, then the need to include the title is minimized, and it represents most of the size of the common information. The size and date should always be included because they can be used as filters and selection criteria. Other information such as number and type of graphics or an abstract might also be included. When at least the title is included in each *surrogate*, the *skeleton* is considered *augmented*. The more information that is included, the more useful the *skeleton* is for truly distributed search where access to the original document set is relatively expensive.

While the resulting *skeleton* can be used directly for visualizations or other presentations to the user, it is useful to introduce an intermediate step that applies filters to the *skeleton* to create a more specialized data representation called a *metagraph*. It is called a *metagraph* because the most interesting operation is to restrict interest to a proper subset of the total vector space, and the resulting data set represents a graph in S dimensions, where S is the number of dimensions in the subset space. It is also possible to require that all of the documents in a *metagraph* meet some other criteria, such as containing reference to one or more topics that are not part of the vector subset. The result can be a drastically smaller *metagraph* than the *skeleton* from which it was derived.

Metagraphs, whether they are proper subsets of the *skeleton* or not, can be processed in the virtual memory of modern desktop computers to provide a variety of useful visualizations and analysis. For example, by picking any three topic values of those in *metagraph*, a Cartesian graph of the documents can be presented to the user as described below.

7. DISTRIBUTING SUMMARIES

The key to distributing searching to the desktop is identification of subsets of document set information that can be moved to the desktop. The subset must be small enough to be transported to the desktop economically yet large enough to provide useful search information. The economic issue is that the cost of transporting the information (which largely depends on the size of the distributed information) must correspond to the value it offers for searching. Typically, full-text indexes aren't economic to distribute because they are so large (i.e., 50%-300% of the size of the original document set), and the necessarily textual queries and responses that are supported can be performed just as well on a server.

There are two criteria for effective distribution, and both depend on size. First, the distributed representation of the document set must fit on the user's disk. This depends on the perceived economic benefit of the search information. The representation of a medical journal back-issue set for a researcher might warrant storage of several gigabytes, an easily achieved number on the desktop today. The representation of an automobile document set to a hobbyist might warrant only a few dozen megabytes. The second criterion is delivery bandwidth. Only the smallest representations can effectively be downloaded over a modem link, but direct connections to the Internet can allow hundreds of megabytes to be downloaded if sufficient economic benefit is expected. For representations that aren't time-critical, the mailing of CD-ROMs or tape cartridges make multi-gigabyte representations feasible.

One of the first mechanisms for distributing search was *glimpse*[5], which provides limited text searching of document sets. By removing positional information about terms in a document, a *glimpse* representation of a document set can be as little as one percent of the original set size. (Note for comparison purposes that different criteria were used for measuring *glimpse* indexes than I use for a *skeleton*. *Glimpse* indexes are measured in optimal binary form with no additional information. I have not explored optimal coding of *skeletons* [I use ASCII], and I always include additional information such as title, size and date. I estimate that an augmented *glimpse* index would be as much as five times larger.) *Glimpse* is primarily used in conjunction with the *harvest* data distribution system, so it has not seen much use outside of academic environments. I mention *glimpse* in some detail because its characteristics make it highly complementary to the system I am proposing.

I propose that *skeletons* and *metagraphs* can be distributed economically. Since a typical *skeleton* is about one percent of the original document set size, it can be distributed to desktops as readily as a *glimpse* index. For example, a one-gigabyte *skeleton* could be used to explore a 100 GB document set.

If it is not effective to distribute a *skeleton*, the selection of a subset of dimensions allows the distribution of a *metagraph* specific to a specific users interest. Whereas 300 or a thousand dimensions might characterize a skeleton, a *metagraph* might only include ten or twenty chosen dimensions. Depending on the distribution of documents and the application of additional filters, a metagraph might be one-tenth the size of the *skeleton*. In practice, a *metagraph* can be anywhere from one to 100 percent of the size of the *skeleton*. Fortunately, it is easy to determine the size of a *metagraph* before downloading it, allowing adjustment of its creation to optimize the tradeoff between size and capability.

Finally, the graphic representations resulting from queries can be downloaded. These graphic representations will be very information dense, showing hundreds or a thousand documents that result from the query. The graphic image will typically be a few hundred thousand bytes, allowing easy download while still containing enough information to allow interactive exploration on the desktop.

An important point about distributed search is the tension between freedom of access and proprietary interest. Many document sets are considered quite valuable by their owners, and access to them is controlled to reduce the chance of economic loss. Often even search access must be controlled for the same reason. Note that in a *skeleton* neither the *reduction rules* (except the topic names) nor any real content of the original document set remains. Thus, the distribution of *skeletons* (or *metagraphs*) allows desktop searching of proprietary document sets without any chance of loss of proprietary value. This might open up many new document sources for casual use, since the user would only have to pay for the desired documents, not the right to look for them.

The image I am trying to create is an environment in which users can download to the desktop the appropriate level of information. Depending on need and purpose, a user could choose among the following levels of distributed summaries of a document set:

- The full document set, if not proprietary. (N megabytes)
- A full-text index of the document set, if not proprietary. (N megabytes)
- An augmented *glimpse* index of the document set, if not paranoid proprietary. (N/20 megabytes)
- An augmented *skeleton* of the document set. (N/50 megabytes)
- An unaugmented *glimpse* index of the document set, if not paranoid proprietary. (N/100 megabytes)
- An unaugmented *skeleton* of the document set. (N/250 megabytes)
- An augmented *metagraph* of the document set. (N/1000 megabytes)
- An unaugmented *metagraph* of the document set. (N/5000 megabytes)
- A VRML graph of a query response. (250 kilobytes).

The "paranoid proprietary" above refers to the fact that document terms appear in the *glimpse* index, though in general they provide insufficient information to reconstruct document contents. In rare cases of content or if the document set owners want to be extra careful, *glimpse* indexes might not be made available for this reason.

A fully automated system could determine the optimal level of download for the user, based on his preferences, resources and usage patterns.

8. SEARCHING ON THE DESKTOP

Once a *skeleton* or *metagraph* is on the user's desktop, the user can search without further access to the network or the original source of data. The manner in which queries are given and in which responses are presented to the user depends on the user's resources and requirements. I will describe here a 3D graphic presentation that requires considerable computing power and 3D graphics performance.

The advantages of a 3D graphic presentation of response data include:

- A document can be identified by multiple levels of detail depending on focus.
- The simplest representation of a document can be small and simple so that large numbers (> 1000) can be represented in a single display.
- Document representations can be organized to aid understanding and use of query result.

The specific presentation I have chosen is graphing in a 3D Cartesian coordinate system. Each of the three axes provides the strength of the document in one of the dimensions of the query. Each document is represented in the graph by a small cube with no apparent features. The cube is placed to show the document's topic strength relative to each of the three axes. When the cube is brushed (touched by the cursor), the date and title of the document are displayed. When the cube is selected (mouse click during brush), the document is retrieved according to retrieval rules built into the *skeleton*.

A query in the proposed system consists of a logical combination of topics. The topic evaluations can be treated as binary filters (zero and non-zero) to provide normal filter equations. The combination of carefully chosen topic profiles and carefully chosen thresholds (the expert contributions) ensures that most documents are not about most topics, so the evaluations act as very effective filters. I found in working with the Tipster document collection that the logical *and* of three topics would typically select just a few hundred documents out of a total of 183,000.

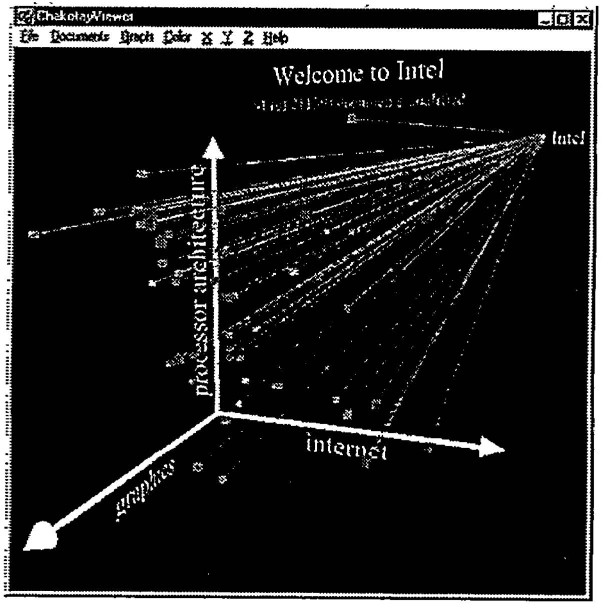
A query in the proposed system has two important characteristics. First and probably most important is its ability to act as a filter. As described above, this allows focusing on a relatively small number of documents out of a very large set. The second is the numeric value that purports to indicate topic strength. While the topic value certainly doesn't correspond completely to an expert evaluation of the same document, I believe it is useful enough for the problem at hand, which is graphing the documents in a 3D Cartesian coordinate system.

A query consisting of the logical *and* of three (or more) topics can be used to create a graphical query response. The query results are graphed using each of the (first three) topics as a graph axis. Topics in excess of three can be indicated with connecting lines as will be shown later. When the natural query consists of less than three topics, objective values such as document size and publication date can be used to round out the query. (As a minor point, thresholds can be applied to the objective criteria to make them effective filters as well. For example, only include documents published between two dates.)

The Cartesian graph is not intended to be an output of the system. It is intended to communicate query results to the user as part of an interactive query cycle. The system doesn't expect that any single well-formed query will provided the answer. Current query methods seem to assume that, and the only interaction they support is a process of converging on the "golden query" that will get the desired answer. The proposed system, because it works on the desktop, where large query state can be maintained, involves the user more intimately in the following ways:

- The user can apply various filters to the document set, allowing the user to limit processing and graphing to a proper subset of the total set.
- The user can interactively specify that individual documents or groups of documents that appear in a graph be earmarked for later processing. This is analogous to a calculator memory.
- The user can collect earmarked documents from several queries to be used for later processing. This is analogous to adding to the calculator memory.
- The user can interactively specify that individual documents or groups of documents that appear in a graph be ignored during subsequent queries.
- The user can choose the domain of a query. For example, a query can be applied to the entire document set, a prefiltered subset, the documents displayed in the most recent graph, or the contents of the calculator memory.
- At any point the user can look at the titles of several documents. I have found the immediate display of date and title information in response to a brush to be an effective mechanism for examining the titles of many documents.
- At any point the user can look at the contents of a document. This depends, of course, on availability and response time of the document set. For Web pages the response time is that of the underlying browser.
- Typically, the first document of interest is the one furthest from the origin of the graph, since it is strong in all three topic coordinates. The next document to look at depends on the interest of the user. The result of a traditional query is a single list in a somewhat arbitrary order. The Cartesian graph gives the user cues to use in choosing documents for examination.
- When a query fails for lack of appropriate contents, traditional query systems will typically still return some responses that passed some arbitrary threshold. In the proposed system, even if some marginal documents are returned, the failure will typically be evident to the user because of the distribution of documents in the graph.

The following image is typical of those provided by the system. It graphs all the documents in the Intel Web site that discuss the four topics "graphics", "internet", "processor architecture". Since more than four topics were provided, the first three were used as the graph axes, and the existence of the word Intel is indicated by the lines connecting word in the upper right corner.



Additional visual cues are available to the user. Color can be used to indicate the source of the document or its possible importance.

When the size of the *skeleton* is significantly larger than the amount of RAM available, it will be necessary to filter the document set using a non-graphical query, creating a *metagraph*. This step can be ignored if enough RAM is available. The non-graphical query will usually be an enumeration of a subset of interesting topics. When this doesn't make the *metagraph* small enough, one or more topic (or date) filters can be applied. The processing required to create a *metagraph* from a *skeleton* is very fast, requiring only a single pass through the *skeleton* with minimal processing on each document *surrogate*.

The most significant limitation of searching in the new system is that only the predefined topics (and objective values such as size and date) can be used as query terms. This makes detailed queries difficult or impossible, depending on how clever the topic expert was, but the selection of topics and careful topic definitions can make casual or initial searching very effective. If scanning the results of visual queries doesn't find the desired documents, further search steps may be required. Ideally, the new system could be used to narrow down a search so that a more specific search could be used to find specific documents. Unfortunately, most current search methodologies don't accept a document subset as input, for all the reasons discussed earlier. Because it is also desktop capable, *glimpse* could be used to fill this role.

9. CURRENT STATUS

At the current time a prototype implementation of the system has been built in C++:

- It is currently targeted at two document sources: the Tipster CD-ROM and the Intel Corporate Presence Server (CPS, Intel's public World Wide Web site).
- Each target has a different front end (for reading the data), different *reduction rules*, and different document access mechanisms. They share all other components of the system.
- Two interaction models have been prototyped: a Common Gateway Interface (CGI) model that allows Web presentation, and a Microsoft Foundation Class (MFC) model that allows standalone operation.
- I have written a fairly complete *reduction rule* set for the Intel CPS. The *rule* set for Tipster is currently incomplete.
- The prototype formats for the intermediate files (*skeleton*, *metagraph*, etc.) are coded in ASCII and otherwise unoptimized.
- Additional features not described in the paper, such as recognizing and drawing hypertext links, have been implemented.

Most of my work to this point has been creating an entirely new infrastructure to support the new system. Besides the obvious need to optimize and systematize the prototype, the primary area of investigation needs to be in effective visualizations. Extensions and variations of the Cartesian graph have been considered but not explored, and other types of visualizations, both of documents and the document set as a whole, have not really been considered. [Since writing this paper, a robust visualization has been implemented in OpenGL.]

And of course, the effectiveness and usefulness of the new system in various problem areas has not really been tested or evaluated.

10. CONCLUSION

While the proposed system violates many of the fundamental premises of current IR research, it meets head-on all of the problems described at the beginning of the paper. Further development and testing will be needed before we fully determine its worth, but the importance of the problem area makes that work worthwhile.

Considerable barriers face the widespread use of the new system. They include:

- user unfamiliarity with interactive search and graphical presentation,
- the requirement of expert authorship,
- lack of widespread availability of desktop computers of adequate power,
- lack of infrastructure for regularly moving large intermediate files, and
- lack of integration with other search technologies.

Separate progress is being made in several of these areas. I hope that my work can help foster progress in the others. I doubt that search in the 21st century will consist of entering words and reading a list of results.

11. ACKNOWLEDGMENTS

The document-handling infrastructure described here is based on other work in *relevance* technology by my colleagues in the Information Architecture group of Intel Architecture Labs. The visualization work has certainly benefited from the inspiration provided by work from Xerox PARC.

LEXIS and NEXIS are registered trademarks of Reed Elsevier Properties Inc. DIALOG is a service mark of Knight-Ridder Information, Inc.

12. REFERENCES

1. Ahlberg and E. Wistrand, "IVEE: an information visualization & exploration environment," *Information Visualization '95 Proceedings*, IEEE Computer Society Press, pp. 66-73, 1995.

2. Arents and W.F.L. Bogaerts, "Concept-based retrieval of hypermedia information: from term indexing to semantic hyperindexing," *Information Processing & Management*, Vol. 29, No. 3, pp. 373-386, 1993.

3. Faloutsos, "Access methods for text," *ACM Computing Surveys*, 17 (March 1985), pp. 49-74.

4. Korfhage, "To see, or not to see – Is that the query?," *Communications of the ACM*, 34, pp. 134-141, 1991.

5. Manber and S. Wu, "GLIMPSE: a tool to search through entire file systems, " *Winter USENIX '94 Technical Conference Proceedings*, 1994.

6. Wise, et alia, "Visualizing the non-visual: Spatial analysis and interaction with information from text documents," *Information Visualization '95 Proceedings*, IEEE Computer Society Press, pp. 51-58, 1995.

7. Witten, A. Moffat, and T.C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, Van Nostrand Reinhold, New York, 1994.