# Sorting out Searching
## A User-Interface Framework for Text Searches

**Ben Shneiderman**
Department of Computer Science and Human-Computer Interaction Laboratory,
University of Maryland, College Park
(ben@cs.umd.edu)

**Donald Byrd and W. Bruce Croft**
Center for Intelligent Information Retrieval, University of Massachusetts at Amherst
(dbyrd, croft@cs.umass.edu)

June 1998

Abstract: Current user interfaces for textual database searching leave much to be desired: individually, they are often confusing, and as a group, they are seriously inconsistent. We propose a four-phase framework for user-interface design. The framework provides common structure and terminology for searching while preserving the distinct features of individual collections and search mechanisms.

---

This paper appeared in *CACM* 41, 4 (April 1998), pp. 95–98. An earlier but more detailed version appeared as Tech Report IR-107 in this series, and online in *D-Lib Magazine:*

http://www.dlib.org/dlib/january97/01contents.html

This material is based on work supported in part by the National Science Foundation, Library of Congress, and Department of Commerce under cooperative agreement number EEC-9209623,by NRaD Contract Number N66001-94-D-6054, by NASA contract NAG-528-95, and by NSF contract IRI-96-15534.

Any opinions, findings, conclusions and/or recommendations expressed herein are those of the authors and may not reflect those of the sponsors.

## Introduction

**The problem.** We see an opportunity to improve dramatically the user experience for textual database searching. The ideal user interface is comprehensible, predictable, and controllable, but many of the current text-search interfaces—especially on the World Wide Web—are unnecessarily complex and obscure key features. The result is confusion and frustration for advanced users as well as beginners, for scientists as well as students (Somerson, 1996).

Improved user-interface design is part of the solution, but even then, as users move from one search system to another, inconsistencies can cause mistaken assumptions and increase failures to find relevant documents. For example, the search string "Hall effect" could produce (among many other possibilities) a:

- exact-match search for "Hall effect"
- case-insensitive exact-match search for "hall effect"
- best-match search for "Hall" and "effect"
- Boolean search for "Hall" AND "effect"
- Boolean search for "Hall" OR "effect"

Few systems indicate clearly the interpretation they are using. Furthermore, systems often use surprising query transformations, unpredictable stemming algorithms (see under "Variants" below), and mysterious weightings for fields. Finally, in many systems the results are displayed in a relevance ranking whose meaning is a mystery to many users (and sometimes a proprietary secret).

**Towards a Solution.** We propose a four-phase framework for increasing clarity and user control while reducing inconsistencies in text-search user interfaces. The automobile user interface is something we now take for granted, but it took decades to reach the current level of refinement and standardization (Oliver and Berkebile, 1968)—and remaining inconsistencies like left/right variations from country to country still cause serious problems for travelers. We believe that by cooperation in interface design we can spare text-search users millions of conceptual fatalities.

## The Four-Phase Framework for Search

The four-phase framework gives great freedom to designers of specific systems to offer a variety of features in an orderly and consistent way. The phases are: **formulation** (what happens before the user starts a search); **action** (starting the search); **review of results** (what the user sees resulting from the search); and **refinement** (what happens after review of results and before the user goes back to formulation with the same information need).

Users begin the search process by considering their information needs and clarifying their search goals. Then users are ready to employ a computer-based system for the four phases:

**1. Formulation:** This is the most complex phase in that it involves multiple levels of cognitive processing and decisions of several types. These decisions include the *sources* of the search, i.e., where to search; which *fields* of documents to search; what *text to search for;* and what *variants* of that text to accept. Some systems walk the user through these decisions, but they cannot always be made in a predetermined order; nor do they exhaust the query-formulation possibilities.

**a. Sources:** The first step in performing a search is normally to decide where to search. This is often a single physical database, but increasingly it is multiple and distributed databases, accessed across a network.

Even if technically and economically feasible, searching all libraries or all collections in a library is often undesirable. When users know where the relevant material is, they generally prefer to limit their searches to that library, collection, or range of documents.

**b. Fields:** Each document in a collection may have multiple fields (sometimes called attributes, components, or tags). Users may wish to limit their search to specific fields, or to give a higher rank to documents whose titles contains search terms: see for example the elaborate weighting algorithm used by THOMAS (Croft, Cook, and Wilder, 1995).

Searches may also be restricted by structured fields (year of publication, language, publisher, etc.).

**c. What to search for:** There are various ways to express what to search for in full text; the most important are probably (1) unstructured text, (2) text with embedded operators, and (3) text with operators specified separately. Pure unstructured-text interfaces are unusual: most of the popular Web search services and other systems such as INQUERY accept either unstructured text or text with embedded operators. An example of the latter is the widely-used syntax of “"city guide" +Boston” (the words “city” and “guide” must appear nearby, and the word “Boston” is required). Finally, some systems—for example, HotBot and Open Text—offer text with separate operators. All three ways can be effective, but only if they are used properly.

In many situations, searches on meaningful phrases are much more effective than searches on the words of the phrase. For example, for someone searching for information on air pollution, the phrase “air pollution” is likely to find many fewer irrelevant documents (higher precision) than the pair of words “air” and “pollution”—though it will tend to overlook relevant documents that refer to “air quality” or “atmospheric pollution” (lower recall). It should be easy for users to specify that a series of words should be considered a phrase.

The unstructured-text approach, often called “natural language”, can be very misleading. For example, many systems treat “and” and “not” as stop words. In such a system, the query “bees and not honey” means the same thing as just “bees honey”: compared to the query “bees”, it is *more* likely to retrieve information about honey, not less. The traditional solution to such ambiguities is feedback to users on how the system interpreted their queries, but it is difficult to clarify the issues for novices.

In theory, the text-with-embedded-operators approach can be completely unambiguous, and it can handle phrases and fields. However, it is well-known that many users have trouble with this approach. One reason is lack of standardization—the syntax and meaning of embedded operators vary considerably from one system to another, so it is easy to get confused. Another problem is the danger of inadvertent activation: innocently using text that the user thinks of as unstructured, but which contains characters or strings that will be interpreted as embedded operators. For example, in AltaVista advanced search, "*" indicates a "wildcard" (matching anything); in Excite! and HotBot, it has no special meaning. These two problems are related in that lack of standardization can confuse users and lead to inadvertent activation. To experience these problems and others, just use several Web search services to look for information on the electronic-commerce company E*Trade.

Another way to specify phrases unambiguously is text with operators specified separately: the program considers the contents of every text-entry box as a phrase, and clearly says so on the screen. Then multiple entry boxes must be provided to allow for multiple phrases. Of course, a text-entry box must also accept a single word. If choices of Boolean operations, proximity restrictions or other strategies for combining the boxes are available, then users should be able to express them; regardless of whether any choices are available, users must be told what combining technique is being used. Ideally, users and/or service providers should have control over stop lists (common words, single letters, etc.); at a minimum, users should be gently warned when they try to search for a stopword.

The basic issue is: Does the program interpret the query the way the user intended it, and—even if it does—does the user *know* that the program interprets it that way?

**d. Variants:** Users are often unsure of the exact value of the field they want; indeed, there may not be any single value that is appropriate. As a result, users may want variants to be accepted. In structured fields of text databases, as in traditional databases, this may include a range on a numeric or date field. In unstructured text fields, interfaces may allow user control over:

- capitalization (case sensitivity)
- stemmed versions: searching for "teacher" finds words like "teach" and "teaching"
- partial matches: searching for "biology" retrieves "sociobiology" and "astrobiology"
- synonyms: searching for "cancer" finds "malignant tumor"
- abbreviations/acronyms: searching for "Digital Equipment Corporation" finds "DEC"
- stop words: common words such as "the" or "to" are excluded.

Other possibilities include phonetic variants, e.g., from N-grams or soundex-like methods, and broader or narrower terms, presumably from a thesaurus.

In all cases, the user interface should make it clear how variants are handled.

**2. Action:** Searches may be started explicitly or implicitly. The typical process in current systems is for users to click on a Search button to initiate the search and then wait for the results. But an appealing alternative is "dynamic queries": there is no Search button but the result set is continuously displayed and updated as the search is changed. Research prototypes such as the FilmFinder and HomeFinder (Shneiderman, 1994) and commercial systems such as Folio Views apply this technique. It requires adequate screen space and rapid processing, but the advantages are great: users can broaden, narrow, or refocus their search several times in as many seconds.

In situations where it is not practical to re-run the query and update results continuously—for example, when the database and the user are connected by a network with limited bandwidth—the "query preview" approach is worth considering (Doan et al, 1996). In this approach, changes to the query simply update a display (perhaps just an estimate) of the number of hits. The query is not actually run until the users request the full results, presumably when they are satisfied that the number of hits is neither zero nor so high as to be cumbersome.

**3. Review of results:** Many information retrieval interfaces let users specify result set size (for example, a maximum of 100 documents), contents (which fields are displayed), sequencing of documents (alphabetically, chronologically, relevance ranked,...), and, occasionally, clustering (by field value, topics,...). All of these capabilities can be valuable, but they all simply try to make a list of documents easier to handle. A query against a large database, even a query that is well focused, can produce so many potentially-useful hits as to be overwhelming—say, several hundred or more. Fortunately, much more can be done to display results in a useful form.

Recent work in information retrieval interfaces, capitalizing on general information-visualization research, has dramatically expanded the palette of display techniques. For example, LyberWorld (Hemmje et al, 1994) displays document icons inside a circle, with terms around the circumference "pulling" the documents towards themselves; the terms can be moved and the strengths of their pulls varied. Rao et al (1995) describes such techniques as tilebars, perspective walls, cone trees, and document lenses.

Search interfaces should also provide helpful messages to support progressive refinement. For example, if a stop word or misspelling is eliminated from a search input window, or stemmed terms, partial matches, or variant capitalizations are included, users should be made aware of these changes to their query.

**4. Refinement:** One of the most important ways in which current information retrieval technology supports progressive refinement is relevance feedback. An empirical study confirms that users produce superior searches and are more satisfied if they can see and manipulate the words relevance feedback adds to their query (Koenemann and Belkin, 1996). Another aspect of refinement is supporting successive queries. As searches are made, the system should keep track in a history buffer to allow review, alteration, and resubmission of earlier searches.

## Conclusions

In summary, the four-phase framework focuses on:

**1. Formulation:**

- **Sources:** specify which libraries and/or collections to search and the search range within them.

- **Fields:** specify which text fields are to be searched. Searches may also be restricted by structured fields.

- **What to search for:** text selected or typed by user, perhaps as one or more phrases.

- **Variants:** control over features like case sensitivity, word stemming, partial matches, phonetic variants, stop words, synonyms, abbreviations, and broader or narrower terms. In all cases, the user interface should make it clear which variants, if any, are allowed.

**2. Action:** how does a search get initiated—explicitly (e.g., with a button), or implicitly (e.g., when some aspect of the query is changed).

**3. Review of results:** conventional options are, for example, to specify result set size, layout, sequencing (alphabetically, chronologically, relevance ranked, etc.), and contents (which parts and fields are displayed). Less conventional interfaces might employ a wide variety of techniques, including many based on information-visualization research.

**4. Refinement:** provide feedback on search results with informative messages and clustering of results. For example, enable progressive querying, especially with relevance feedback; history keeping; and extraction of results to files, perhaps for use in e-mail.

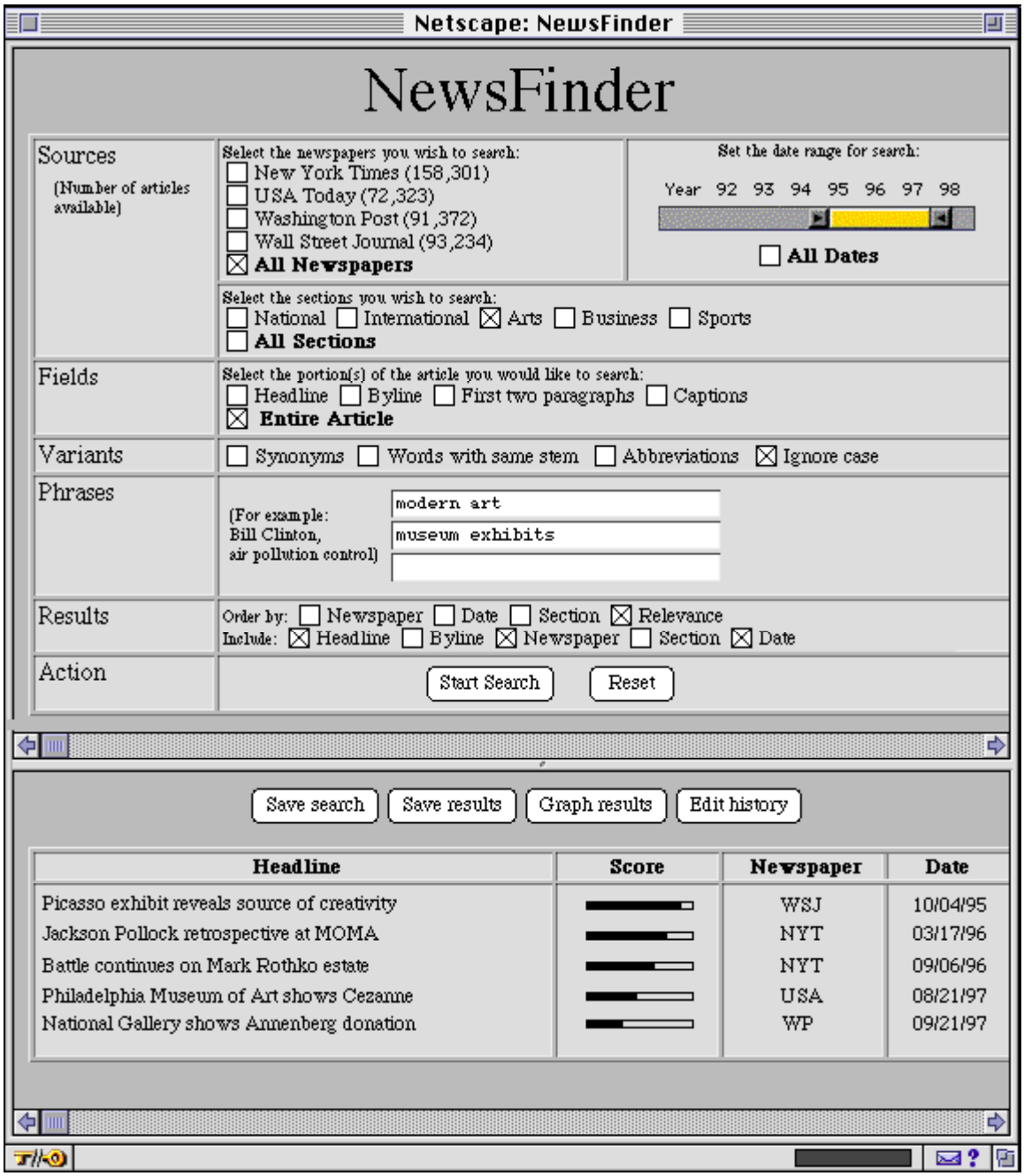


Fig. 1

The sample Web interface shown in Fig. 1 requires nothing more advanced than HTML tables and forms, but in some cases, applying the framework successfully over the Web may require more powerful tools such as Java. On the other hand, while we have thought mostly about text situations, we suspect our framework will prove appropriate for multimedia and some traditional database management applications.

Finding common ground for search interfaces is difficult, but not finding it would be tragic. While early adopters of technology are willing to push ahead to overcome difficulties, the middle and late adopters will not be so tolerant. In particular, the future of the World Wide Web as a universally acceptable tool may depend on our ability to reduce frustration and confusion for the masses of users, while enabling them to reliably find what they need. Some progress has been made; much more remains to do.

## References


Croft, W. Bruce, Cook, Robert, and Wilder, Dean (1995). Providing government information on the Internet: Experiences with THOMAS. *Proc. Digital Libraries 95 Conference,* ACM, New York. Also available at http://csdl.tamu.edu/DL95/papers/croft/croft.html.

Doan, Khoa, Plaisant, Catherine, and Shneiderman, Ben (1996). Query Previews in Networked Information Systems. *Proc. Third Forum on Research and Technology Advances in Digital Libraries, ADL '96 ,* IEEE CS Press, 120-129. Also available as TR 95-16 at http://www.cs.umd.edu/projects/hcil/Research/tech-report-list.html#1996.

Hemmje, M., Kunkel, C. and Willett, A. (1994). LyberWorld - A Visualization User Interface Supporting Fulltext Retrieval. Croft, W.B. and van Rijsbergen, C.J. (eds), *Proc. 17th Annual Int. Conference on Research and Development in Information Retrieval (SIGIR 94)* , Springer Verlag, 249-257.

Koenemann, Juergen and Belkin, Nicholas (1996). A case for interaction: A study of interactive information retrieval behavior and effectiveness. *Proc. CHI 96 Human Factors in Computing Systems,* ACM Press, New York, NY, pp. 205-212.

Oliver, S.H., and Berkebile, D.H. (1968). *The Smithsonian Collection of Automobiles and Motorcycles.* Smithsonian Institution Press, Washington.

Rao, Ramana, Pedersen, Jan, Hearst, Marti, Mackinlay, Jock, Card, Stuart, Masinter, Larry, Halvorsen, Per-Kristian, and Robertson, George (1995). Rich Interaction in the Digital Library. *CACM* 38, 4, pp. 29-39.

Shneiderman, Ben (1994). Dynamic queries for visual information seeking. *IEEE Software* 11, 6, pp. 70-77.

Somerson, Paul (1996). Web Coma. *PC Computing* (August 1996), 57.