ben shneiderman
university of maryland

A Framework for Search Interfaces

Tools, techniques, and concepts to optimize user interfaces.

Ben Shneiderman is one of the most famous pioneers of human-computer interaction research, having invented the concept of direct manipulation and written several of the field's best textbooks. Here he presents a conceptual framework for improving Web search interfaces.
—Jakob Nielsen

SEARCHING TEXTUAL DATABASES CAN be confusing for users. Popular search systems for the World Wide Web and stand-alone systems typically provide a simple interface: users type in keywords and receive a relevance-ranked list of 10 results. This is appealing in its simplicity, but users are often frustrated because search results are confusing or aspects of the search are out of their control. If we are to improve user performance, reduce mistaken assumptions, and increase successful searches, we need more predictable design.

The goals of consistent design can be clarified by a look at early user interface design for automobiles. Early automobile designers offered their own distinct designs for a profusion of controls. Some designs, such as a brake that was too far from the gas pedal, were dangerous. There was also a consistency issue. If your brake was to the left of the gas pedal and your neighbor's car had the reverse design, it might be risky to trade cars. Achieving good design and appropriate consistency in automobiles took half a century. Let's hope we can make the transition faster for Web-search user interfaces.

At present, for example, if users enter the search string "direct manipulation" the results will vary depending on the search engine they choose. They might find

- ♦ a search on the exact string "direct manipulation,"
- ♦ a probabilistic search for "direct" and "manipulation,"
- ♦ a probabilistic search for "direct" and "manipulation" with some weighting if the terms are in close proximity,
- ♦ a Boolean search on "direct" AND "manipulation,"
- ♦ a Boolean search on "direct" OR "manipulation," or
- ♦ an error message indicating missing AND/OR operator or other delimiters.

In many systems, there is little or no indication as to which interpretation was chosen and whether stemming, case matching, stop words, or other transformations were applied. Often, the results are displayed in a relevance-ranked manner that is a mystery to users (and sometimes a proprietary secret).

Although finding common ground will be difficult, not finding it will be tragic.

To coordinate design practice, we suggest a four-phase framework that would satisfy first-time, intermittent, and frequent users accessing a variety of textual and multimedia libraries (see Ben Shneiderman, Don Byrd, and Bruce Croft, "Clarifying Search: A User-Interface Framework for Text Searches," *D-Lib Magazine*, http://www.dlib.org./dlib/january97/01contents.html).

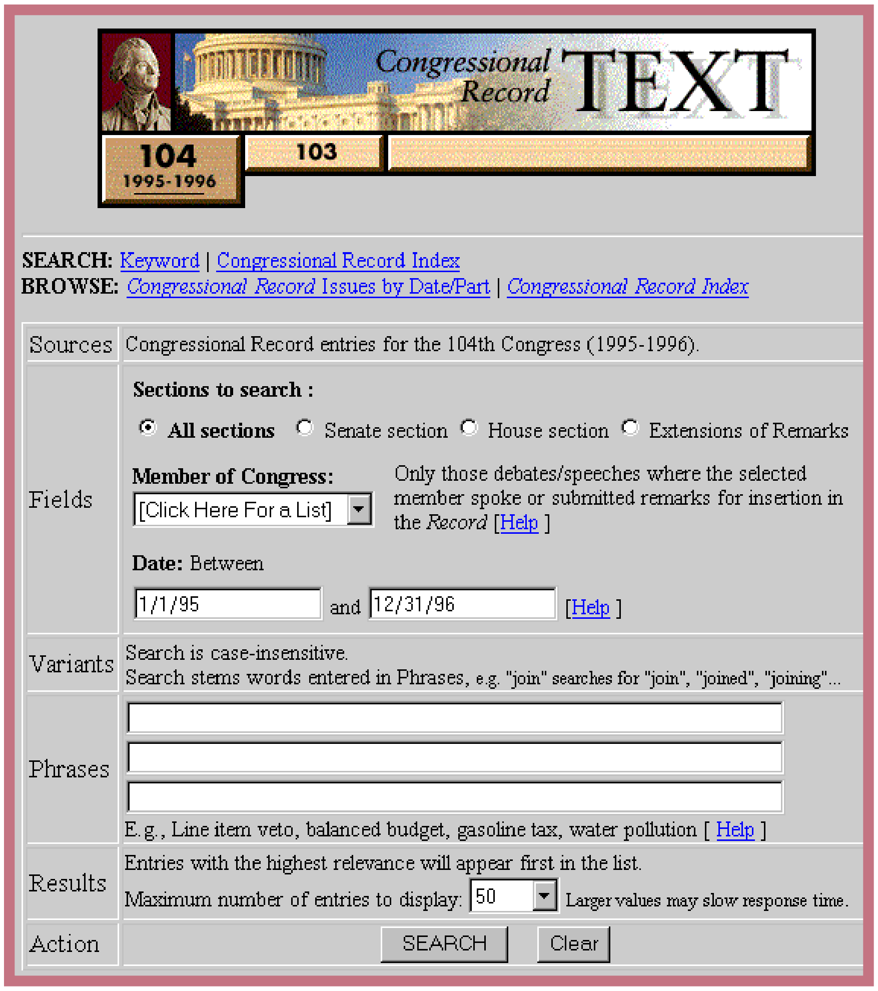
Although finding common ground will be difficult, not finding it will be tragic. Early adopters of technology are willing to push ahead to overcome difficulties; middle and late adopters are not so tolerant. The future of search services on the World Wide Web and elsewhere may depend on how we can reduce user frustration and confusion, while enabling them to reliably find what they need in the rapidly surging sea of information.

FOUR-PHASE FRAMEWORK. The framework has four phases:

1. *formulation*
   - ♦ expressing the search
   - ♦ source of the search
   - ♦ fields for limiting the source
   - ♦ multiple phrases
   - ♦ variants of phrases and terms

Editor:
Jakob Nielsen
SunSoft
2550 Garcia Ave.
MPK 17-305
Mountain View, CA
94043
jakob@eng.sun.com

0740-7459/97/$10.00 © 1997 IEEE



SEARCH: Keyword | Congressional Record Index
BROWSE: *Congressional Record* Issues by Date/Part | *Congressional Record Index*

Sources	Congressional Record entries for the 104th Congress (1995-1996).
Fields	**Sections to search :** [✓ **All sections**] Senate section House section Extensions of Remarks **Member of Congress:** [Click Here For a List] — Only those debates/speeches where the selected member spoke or submitted remarks for insertion in the *Record* [Help] **Date:** Between 1/1/95 and 12/31/96 [Help]
Variants	Search is case-insensitive. Search stems words entered in Phrases, e.g. "join" searches for "join", "joined", "joining"...
Phrases	E.g., Line item veto, balanced budget, gasoline tax, water pollution [Help]
Results	Entries with the highest relevance will appear first in the list. Maximum number of entries to display: 50 Larger values may slow response time.
Action	SEARCH Clear

Figure 1. *An example we developed for the Library of Congress's Thomas system that provides access to the full text of bills in the current and recent Congresses.*

2. *action*
   - ♦ launching the search
3. *review of results*
   - ♦ viewing messages and outcomes
4. *refinement*
   - ♦ taking the next step

As Figure 1 shows, the phases can be used as a framework to design consistent and orderly features.

Formulation. Phase one includes the source of the search, the fields for limiting the sources, the phrases, and the variants.

Even when it is technically and economically feasible, searching all sources is not always ideal. Users might want to limit the source of their searches to a specific library, collection in a library, or subcollection. They might also limit by date, language, or media type, or restrict the search to specific fields, such as titles or abstracts within a collection. Users searching fairly common terms might prefer to retrieve only documents with the term in its title.

Because users often seek items containing meaningful phrases—such as "George Washington" or the "Environmental Protection Agency"—you should provide multiple entry windows for multiple phrases. However, while phrase searches are more accurate than word searches, they can miss relevant items. You should thus offer the option of expanding a search by breaking the phrases into separate words. Phrase searches let users find names (for example, a search on "George Washington" should not turn up "George Bush" or "Washington, DC"). Users should also be able to specify Boolean operations, proximity restrictions, and other combining strategies if possible. Finally, users should have control over stop lists (common words, single letters, and obscenities).

When users are unsure of the exact value of the field (such as the spelling of a city name), they may want to relax the search constraints and accept variations. In structured databases, the variants may include a wider range on a numeric attribute. In a textual document search, interfaces should allow user control over variant capitalization (case sensitivity), stemmed versions (for example, the keyword "teach" retrieves variant suffixes such as "teacher," "teaching," "teaches"), partial matches (the keyword "biology" retrieves "sociobiology" and "astrobiology"), phonetic variants from soundex methods ("Johnson" retrieves "Jonson," "Jansen," "Johnsson"), and synonyms ("cancer" retrieves "oncology"), abbreviations ("IBM" retrieves "International Business Machines," and vice versa), and broader or narrower terms from a thesaurus (the phrase "New England" retrieves "Vermont," "Maine," "Rhode Island," "New Hampshire," "Massachusetts," and "Connecticut").

Action. The initiation of action may be explicit or implicit. Most current systems have a search button for explicit action or for delayed or regularly scheduled actions. The button label, size, and color should be consistent. An appealing alternative is implicit action in which each change to a formulation phase component immediately produces a new set of search results. Dynamic queries, in which users adjust query widgets to produce continuous updates, have proven effective and satisfying. This requires adequate screen space and rapid processing, but the advantages are great.

Results. In the third phase, users get results via messages, textual lists, or visualizations. You can give users control over the size of the result set, which fields are displayed, sequencing (such as alphabetical, chronological, or relevance-ranked), and clustering (by attribute value, topics).

Refinement. Search interfaces should provide meaningful messages to explain search outcomes and support progressive refinement. For example, if a stop word, obscenity, or misspelling is eliminated from a search input window, or stemmed terms, partial matches, or variant capitalizations are included, users should be informed. If the two words in a phrase are not found proximally, then feedback should be given about the occurrence of the words individually. If multiple phrases are input, items containing all phrases should be shown first and identified, followed by items containing subsets. If no documents are found with all phrases, this should be indicated.

There is a fairly elaborate decision tree (around 60 to 100 branches) of search outcomes and messages that must be specified. The system should keep track of searches in a history file so readers can review or repeat earlier searches. Progressive refinement, in which the user refines results by changing the search parameters, should be convenient. Search results and the setting of each parameter should be objects that can be saved, sent by email, or used as input to other programs, such as visualization or statistical tools.

FUTURE GAINS. Designers can use the four-phase framework to give users more control over the search process and make it more comprehensible and predictable. This move is in harmony with the move toward direct manipulation, in which the user can view and control the current system state. Novices may not want to see all components of the four phases initially, but if they are unhappy with the search results, they should be able to view and change them easily.

Some designers will resist any change, but a lively discussion among Web search designers could lead to a refined framework and widespread agreement to apply it. The users will certainly benefit, but search companies will also benefit as users take advantage of their services more often. ◆

Ben Shneiderman is a professor in the Department of Computer Science at the University of Maryland at College Park, where he is also head of the Human-Computer Interaction Laboratory and a member of the Institute for Systems Research. The third edition of his book, Designing the User Interface: Strategies for Effective Human-Computer Interaction, *will be published in July by Addison-Wesley Longman. Shneiderman can be reached at ben@cs.umd.edu.*