A Direct Manipulation Interface for Boolean Information Retrieval via Natural Language Query

Peter G. Anick, Jeffrey D. Brennan, Rex A. Flynn, David R. Hanssen

Digital Equipment Corporation, 290 Donald Lynch Blvd., DLB5-2/B4, Marlboro, MA 01752-0749

Bryan Alvey, Jeffrey M. Robbins

Digital Equipment Corporation, 305 Rockrimmon Blvd. South, CXO3-1/Q3 Colorado Springs, CO 80919-2398

Abstract

This paper describes the design of a direct manipulation user interface for Boolean information retrieval. Intended to overcome the difficulties of manipulating explicit Boolean queries as well as the "black box" drawbacks of so-called natural language query systems, the interface presents a two-dimensional graphical representation of a user's natural language query which not only exposes heuristic query transformations performed by the system, but also supports query reformulation by the user via direct manipulation of the representation. The paper illustrates the operation of the interface as implemented in the AI-STARS full-text information retrieval system.

1. Introduction

The AI-STARS project is an on-going research program at Digital Equipment Corporation, investigating methods for improving full-text information retrieval. Our target audience is Digital's Customer Support Specialists, for whom ready access to on-line technical information is indispensable for quick and accurate handling of a wide range and heavy volume of customer inquiries. Specialists typically must conduct textual information searches while on the phone with the customer and without much time for planning a query. Hence system response time and ease of use are critical to effective use of information retrieval technology in this environment. Our aim is to exploit linguistic and domain knowledge to improve article indexing, query interpretation, and query reformulation.

Our starting point for this research is STARS, the text retrieval system currently in use at Digital's Customer Support Centers. STARS provides full-text retrieval, given a query expressed either as a Boolean expression or as a natural language string. In interviews with STARS users, we found that natural language query is by far the preferred input mode, since users need no training to express topics via the natural language methods of adjectival and prepositional phrase modification, relative clauses, nominal compounds, etc. But, unlike Boolean queries, which provide an explicit semantics defining the set of documents retrieved, natural language queries typically require some "behind the scenes" transformations before they can be matched against a document set. STARS, for example, performs word truncation to remove suffixes, removes noisewords, adds in synonyms, and finally converts the result into a Boolean expression for article matching.

When successful, natural language query is an ideal interface. However, our interviews revealed that natural language query users tend to be confused by system results that don't conform to their expectations of what "should" happen and they often request ways to override (supposed) system default behavior. Thus, the major advantage of natural language query, its avoidance of more cumbersome input languages such as Boolean expressions, is counterbalanced by the need for query-enhancing transformations which typically transpire without the user's awareness.

This state of affairs presents a challenge to projects like AI-STARS, which have the goal of implementing even more powerful "behind the scenes" intelligence. Our approach to this dichotomy is to augment a natural language query facility with a direct manipulation "Query Reformulation Workspace", which incorporates an explicit visual Boolean semantics. This window gives the user a view into the heuristic decisions made by the system and, at the same time, the ability to override, influence, or supplement those decisions.

This paper describes our interface design. We begin with an overview of some of the query reformulation techniques that are appropriate to natural language query systems and describe our implementation of those techniques. We then enumerate a set of design goals we believe a human interface should meet and present a design which comes close to satisfying those goals. We conclude with a discussion of the merits and shortcomings of our approach.

2. Reformulation Techniques

Reformulation techniques can be applied at both article indexing time and at query time to enhance the matching of natural language queries with articles. AI-STARS incorporates the following techniques.

- Stemming. AI-STARS utilizes a lexicon of known words, morphological paradigms, and orthographic rules in order to reduce morphologically inflected forms to their uninflected "citation" forms. This eliminates the need for the user to worry about truncating query terms to increase recall.
- Phrase construction. The lexicon also stores uninflected forms of known contiguous phrases, such as "database management system". This allows the system to recog-

nize known phrases in a user query, obviating the need for the user to explicitly request that the terms be treated as a phrase. Articles are indexed by both the phrase as a whole and the individual components of the phrase.

- Expression canonicalization. Special expressions, such as release version numbers and date expressions, which have multiple surface realizations, are parsed and canonicalized. For example, the variant surface forms "version 5.1-a", "v5.1-a" and "v. 5.1a" are indexed by a single canonical form. This eliminates the need to construct and retrieve on a set of alternative forms at query time.
- Expression generalization. Certain expressions, such as version numbers, encode implicit generalizations. For example, "v5.0a" is a specialization of "v5.0" which is in turn a specialization of "v5". At indexing time, articles containing such expressions are indexed on each generalization of such an expression. This allows for the matching of articles with various levels of query expression specificity.
- Noiseword removal. Words unlikely to contribute to the content of a query, such as articles and prepositions, are automatically excluded from the query.
- Use of thesaurus relations. AI-STARS maintains a database of related terms. Synonyms may be ORed with terms in the query in order to improve recall. More specific terms may be substituted for more general terms to improve precision.

Numerous other reformulation techniques have been investigated in the context of other experimental natural language query systems. [DEBILI88] discusses spelling correction, "explicitation" of implicit concepts recognized via morpho-syntactic frames, and stemmatization relating words with their morphological derivatives. [SALTON75] and [DILLON83] have experimented with phrasal normalization, relating word collocations appearing in semantically similar but syntactically different constructions, such as "retrieval of information" and "information retrieval." [LANCEL88] associates word sequences with semantic filters. [SCHWARZ88] analyzes text into dependency trees reflecting head-modifier relations.

These techniques, while utilizing computational linguistics, stop short of attempting to do a full parse and deep understanding of either the query or text. Researchers [BOGURAEV82, RAU88] have explored this approach to indexing and query interpretation as well. However, our focus here will be on the former kinds of reformulation, those designed to enhance the mapping of a natural language input into a traditional Boolean query. In the STARS full-text system, for example, natural language queries are typically not sentences at all, but rather strings of words and phrases.

3. User Interface Design Goals

Given the context of a full-text information retrieval system which translates natural language queries into Boolean expressions, our aim was to augment the interface to allow the user better understanding of and control over the actual query formulation. We identified a number of design goals for an appropriate human interface for this task.

1. It should make explicit any "behind the scenes" operations done by the system. The results of morphological analysis and other forms of canonicalization should be viewable. If noisewords are removed, this should be made obvious.
2. It should accommodate ambiguity in natural language expressions. Since the system's default interpretation of a query may depend on ambiguity resolution, the interface should have a way to present the ambiguity and indicate the system's response to that ambiguity. For example, in a system which indexes and retrieves on phrases, a natural language input containing a phrase is automatically ambiguous between two interpretations, one matching articles which contain the contiguous phrase, the other matching articles which contain the component terms but not necessarily in contiguous positions.
3. It should provide a natural way of visualizing the Boolean expression created as a result of analyzing the natural language query.
4. It should facilitate query reformulation and experimentation, by making iterative adjustments to the query easy to perform. (e.g. Bates' search formulation tactics [BATES79]).
5. It should integrate smoothly with other information retrieval aids, such as a thesaurus.

4. AI-STARS User Interface

Our experimental realization of the above design goals takes the form of a direct manipulation "Query Reformulation Workspace". This has been implemented using X-windows on a DEC VAXstation as part of the AI-STARS prototype. As mentioned earlier, input to AI-STARS is in the form of a natural language expression. Each word is morphologically analyzed to identify its citation form (the uninflected form as it would appear in a dictionary); meaningful phrases, composed of two or more consecutive words, are recognized; special expressions are canonicalized and generalized; and noisewords are identified. By default, the system makes noisewords, generalizations, and those words that are components of phrases (as opposed to the entire phrases) "inactive", meaning that they are not included in the query executed against the article database. All other terms are initially "active".

The results of this analysis are displayed in the graphical "Query Reformulation Workspace." Figure 1 illustrates the appearance of the workspace after the user has entered the query "Copying backup savesets from tape under v5.0".

All citation forms (for words, phrases, special expressions, generalizations and noisewords) are laid out as tiles in two dimensions, in a chart, or spreadsheet-like, format. The citation forms for the tokens in the original query are displayed horizontally along the top of the chart, thereby defining columns of the chart. Phrases, multi-term special expressions, and ambiguous interpretations of the query terms are displayed below the corresponding items of the top line. The two-word phrase "BACKUP saveset", for example, is displayed in a single tile covering the two columns delineated by its component terms. The tiles of active query terms are indicated by reverse video. The

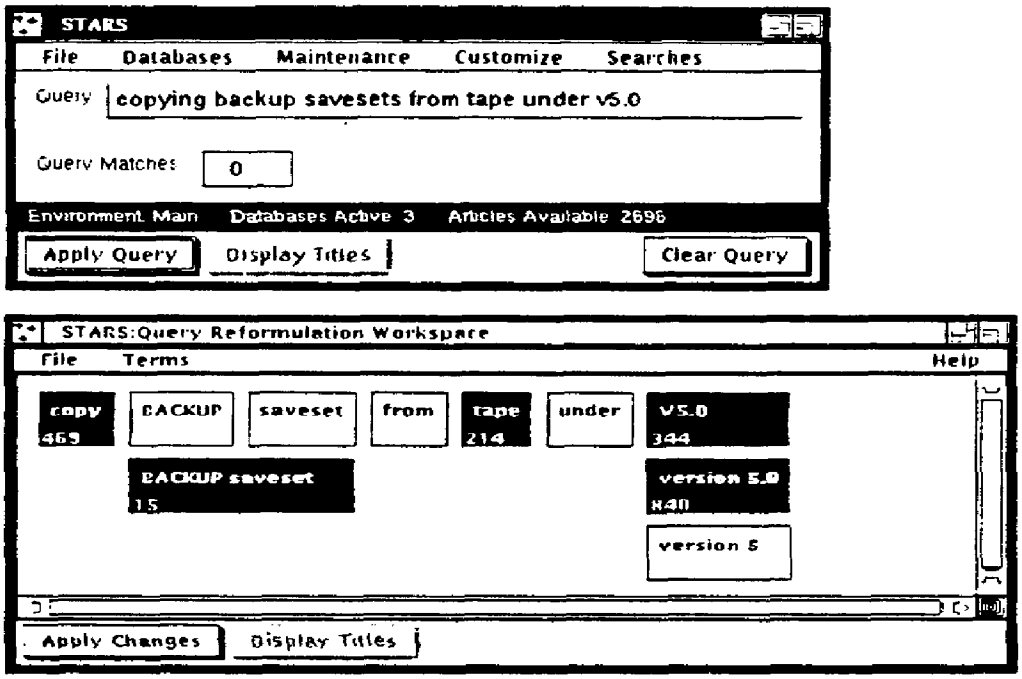


Figure 1 Initial Query

function words, "from" and "under" are inactive. Likewise, the components of the phrase "BACKUP saveset" are inactive, as is the generalized term, "version 5".

The two-dimensional layout provides a simple visual semantics for the Boolean interpretation of a query, while accommodating ambiguity that may exist in a natural language expression. Roughly, tiles which overlap vertically are ORed and those which do not are ANDed. The set of documents retrieved can be described as all those articles containing some combination of terms from any possible left-to-right path through the chart. We will characterize the semantics of the display more completely in section 6.

The Boolean expression corresponding to the displayed configuration of tiles in figure 1 is

("copy" AND "BACKUP saveset" AND "tape" AND ("v5.0" OR "version 5.0")).

As a result of applying the query, the user is informed of the number of articles matching the query and is given the option of viewing a list of their titles. Additionally, the number of postings for each term included in the query is displayed in the lower left-hand corner of the tile corresponding to that term. This feedback can suggest to the user which terms may require broadening or narrowing.

The workspace supports query reformulation via direct manipulation of the display. The system's default choices of active tiles are currently biased towards precision rather than recall. However, each term can be individually included in or excluded from the query by clicking on the tile (which toggles its activation). Figure 2 illustrates the display after the user has deactivated the term "copy" and activated the terms "BACKUP" and "saveset". The Boolean interpretation of this new configuration is

((("BACKUP" AND "saveset") OR "BACKUP saveset") AND "tape" AND ("v5.0" OR "version 5.0")).

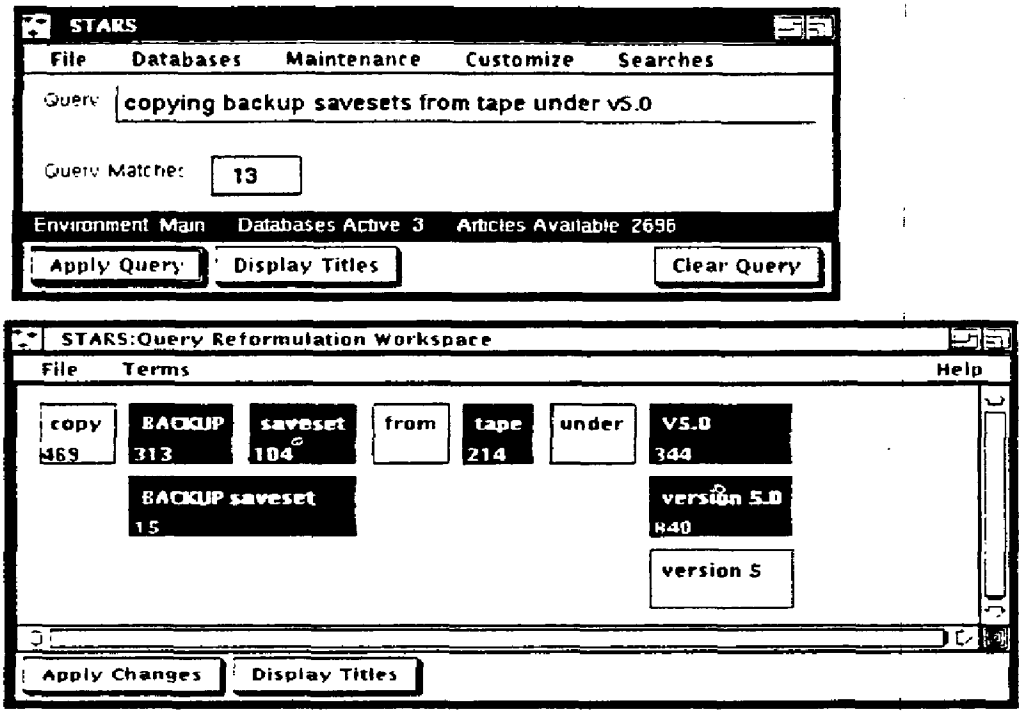


Figure 2 Query after changing activations

In addition to toggling activation of tiles, users can manipulate the search expression further by moving tiles from one column to another via mousing and dragging. For example, in figure 3, the user has moved the "tape" tile into the column below the tile "saveset", changing the configuration to mean

(((“BACKUP” AND (“saveset” OR “tape”)) OR “BACKUP saveset”) AND (“v5.0” OR “version 5.0”)).

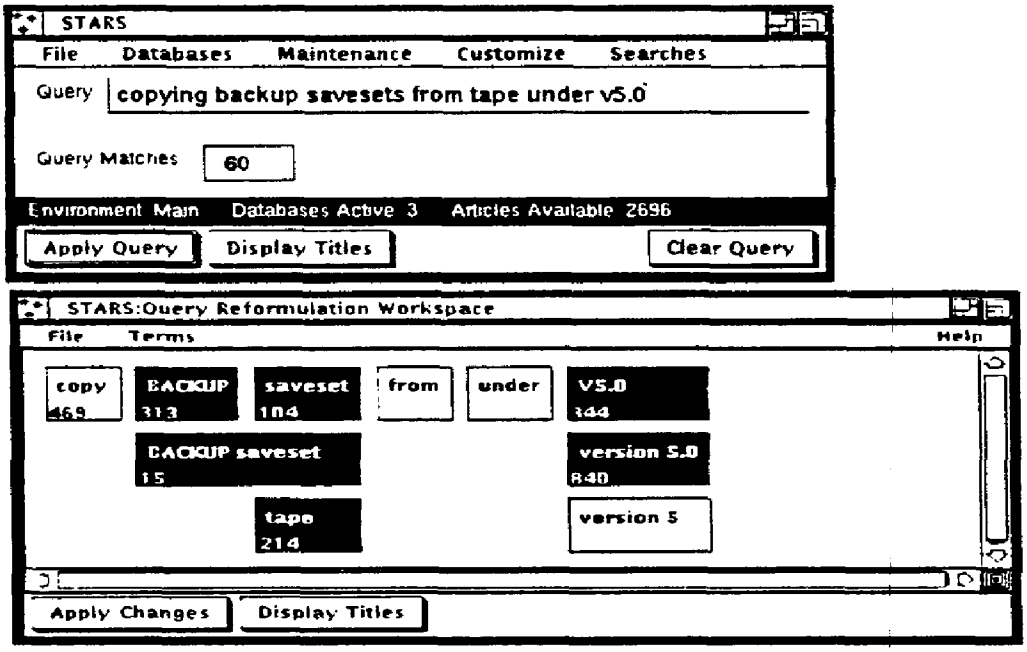


Figure 3 Query after moving tile

Note that the column previously containing "tape" is empty as a result of the move and is no longer displayed.

A tile can be made to extend across more or fewer columns by mousing and dragging the edge of the tile. These operations allow the user complete control over how the terms are to be ANDed or ORed with each other; the Query Reformulation Workspace thus serves as a visually appealing alternative to manipulating Boolean expressions in the form of parenthesized expressions or AND/OR trees.

5. Adding New Search Terms

It is well known that initial user queries rarely contain all the right terms to conduct a successful search [BATES86]. Many systems provide on-line thesauri [FREI83, MCMATH89, THOMPSON89] or support relevance feedback [CROFT86, SALTON89] to supplement an initial query with further search terms.

The AI-STARS system maintains a database of related terms, which are made accessible from the Query Reformulation Workspace. A user can click on a term in the workspace to "select" it and then call on a "related terms" pop-up window to display terms that are related to the selected term in various ways: synonyms, terms containing the term (or a portion of the term) as a substring, phrases containing the term, and conceptually related terms.

In figure 4, the term "tape" has been selected (as indicated by the surrounding bolded rectangle) and the Terms window corresponding to the term "tape" has been displayed to the right of the Query Reformulation Workspace window.

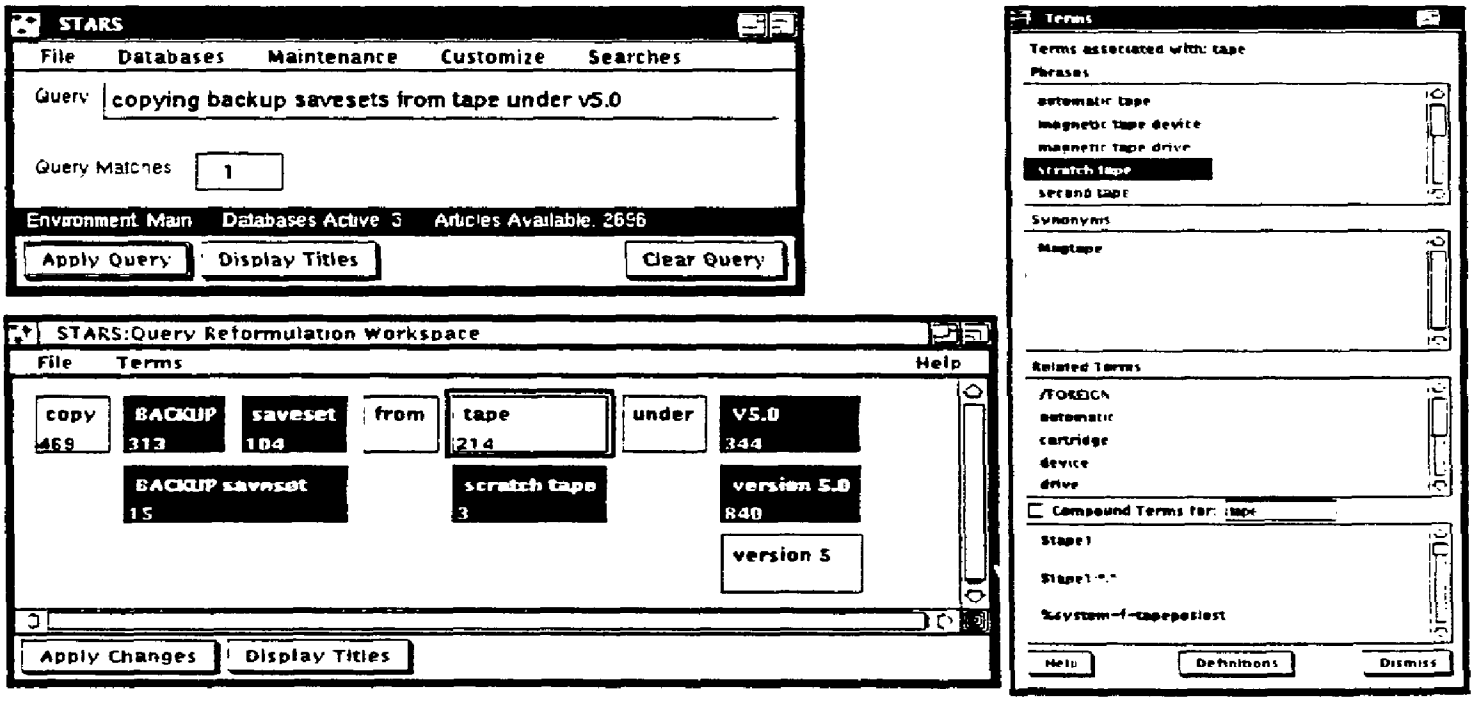


Figure 4 Query after term substituted from terms display

The Terms window is composed of four subwindows, each with its own operational semantics.

- For phrases containing the term, selecting an item from the Terms window has the effect of narrowing the query by substituting the item for the selected term in the workspace. This is achieved by placing the item as an active tile into the same column(s) as the workspace selected term and making the workspace selected term inactive.
- For synonyms, selecting an item from the Terms window broadens the query by adding the item as an active tile in the same column as the workspace selected term, without changing the activation of the workspace term.
- For conceptually related terms, selecting an item narrows the query by appending the item as an active tile after the last occupied column of the chart (visually establishing a new column).
- Compound terms are superstrings of the term which contain at least one non-alphabetic character. Such superstrings are common in a computer science domain, where many terms, such as error messages and facility names, are composed of mnemonic strings embedded in longer strings, using non-alphabetic characters as delineators. When being added to a query, compound terms are treated in the same manner as synonyms.

The association of a default semantics with each Terms subwindow minimizes user effort in augmenting a query, since a user need only select the term to be added, without first considering where the term should be placed in the configuration. In figure 4, the Query Reformulation Workspace has been updated to show the result of selecting the phrase "scratch tape" in the Terms window. The phrase has been placed, as an active term, in the column below the term "tape", which has been made inactive. As each default operation has an immediate visual manifestation, the user is free to adjust the query if, in some circumstance, the default is not what is desired.

As illustrated in the above figures, we separate the natural language query input window from the Query Reformulation Workspace. The user can also add new terms to an existing query by appending them to the original natural language query string. The new terms are processed and added to the display, leaving the earlier portion of the query display intact. This separation into two windows allows the user the option of interacting solely through the natural language query window until the workspace facilities are specifically needed.

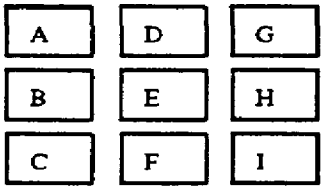
6. Characterization of the Visual Boolean Semantics

The layout for our Query Reformulation Workspace was motivated by the desire to preserve the left-to-right form of the initial natural language input, thereby making it easy for the user to recognize how the tiles in the workspace correspond to the original textual query. The use of the vertical dimension to display alternative terms provides a natural way of representing the ambiguity between a phrase and the components of a phrase taken individually. Yet, as the previous sections illustrate, the set of tile manipu-

lations available permits the construction of arbitrarily complex visual displays. In this section, we describe the visual semantics of the workspace with a series of examples of increasing complexity.

One Column Groups

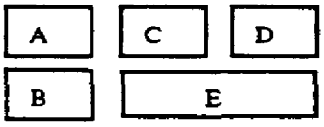
The simplest configurations are those in which none of the tiles span more than one column.



Each column constitutes a group of ORed terms. The ORs formed from each column are ANDed from left to right. This is a standard conjunctive normal form interpretation. We extend the conjunctive form to deal with situations where tiles have been deactivated. If a single tile in one of the columns above is inactive, it simply does not participate in the OR for its column. If all of the tiles in one column are inactive, we have chosen to ignore that column in the query. (The alternative choice would have been to make the query unsatisfiable.) A common situation producing inactive columns is the deactivation of function words, such as "from" and "under" in Figure 1.

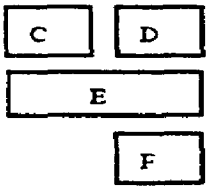
Multi-column Groups

A multi-column group exists whenever a single active tile spans more than one column.



In this example, we consider tiles C, D and E to form a single multi-column group. Within this group, C and D do not overlap, but together they overlap completely with E. We therefore translate this group into the Boolean expression ((C AND D) OR E). The expression for this group is then included in the overall query as if it were a single-column group (the resulting query is [(A OR B) AND ((C AND D) OR E)]).

There are some multi-column groups where tiles are not necessarily displayed in the order in which the expression is generated.



In this configuration, D and F overlap completely, so they are ORed together before ANDing with C (and ORing the result of that with E). So, the resulting expression for this group is ((C AND (D OR F)) OR E).

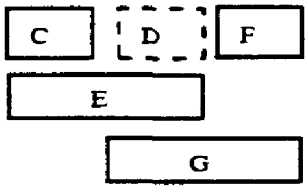
Multi-column group with partial overlap

Sometimes some of the tiles in a multi-column group are inactive.

By analogy to the situation in which a column contains no active tiles, there are two ways the space left by tile D can be interpreted. Breaking down the group into two subgroups (C and "D" as one subgroup, and E as the other), either the subgroup containing C cannot satisfy the query on its own, or it can. We have chosen the latter interpretation, i.e. to ignore the "column" occupied by D in the subgroup, so the Boolean interpretation for the whole group ends up being (C OR E).

Multi-column group with complex overlap

There are certain overlapping situations which are much harder to interpret.



In this situation, we extend the notion of a group to include all of the tiles above. A more formal definition of a group is that it consists of all the active tiles in a series of adjacent columns, where each of the boundaries between these adjacent columns has at least one active tile spanning it, but the boundaries on either side of the the outermost columns in the group have no tiles spanning them.

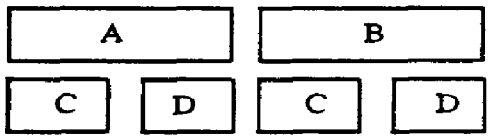
The major question for such a group is how we would like it to be interpreted. It is clear that an article containing both E and F should satisfy the query. Likewise, an article containing both C and G should satisfy the query. It is not clear whether an article containing C and F, but neither E nor G, should satisfy the query. The algorithm which we use (and which is described in section 8) will allow this last possibility to satisfy the query. We believe that this is consistent with how we have interpreted the other configurations containing inactive tiles. Nevertheless, such situations are difficult to analyze.

Fortunately, such circumstances rarely occur. Where two terms overlap each other in the way that E and G do, they tend to represent competing concepts. For example, if the user typed in "operating system management," the system might know about the phrases "operating system" and "system management." It is unlikely that the user would want to query on both these concepts simultaneously.

7. Explicit Boolean operators

The visual Boolean semantics described above allow the expression of any propositional calculus expression formable using the connectives AND and OR. Even

so, there are queries that cannot easily be expressed in our two-dimensional graphical language, such as ((A and B) or (C and D)). Visually representing this query would require repetitions of the same terms at different points along the horizontal axis, as shown here:



Furthermore, the traditional Boolean NOT operator is not available. While these are potential shortcomings of this approach, they may have little practical consequence. McAlpine and Ingwersen [MCALPINE89], for example, limit their query-by-forms search expression interface to conjunctions of ORed terms, arguing that this is sufficient in most cases.

Nevertheless, we are experimenting with the addition of an optional "escape hatch" to full propositional calculus, by allowing the user to include the Boolean operators AND, OR, and NOT directly in their natural language queries. When one of these terms is encountered in a query, it is treated as ambiguous between the English word and the Boolean operator, with the Boolean interpretation active by default. With the use of parentheses, this permits the formation of arbitrary Boolean expressions. Natural language sequences appearing between Boolean operators are interpreted as above.

8. An Algorithm for Interpreting Chart Configurations

We implement the visual Boolean semantics described in section 6 by employing a recursive divide-and-conquer strategy. Applied to any subgroup of terms generated from the tiles in the workspace, it generates a tree of nodes, where the internal nodes are Boolean query operators (AND, OR and NOT), and the leaves are the workspace terms. The "divide" part of the algorithm splits a group of terms into multiple smaller groups of terms, to which the algorithm can be re-applied. The "conquer" part groups multiple subtrees together into a single larger tree by supplying a new AND or OR internal node. Once a tree is generated for all the terms in the workspace, it can be interpreted directly to search the database of articles.

The same algorithm handles both the default workspace semantics and explicit Boolean terms as described in the previous sections. The process of splitting and regrouping terms recursively into a tree is not complicated for explicit Boolean terms, and is handled at the top levels of recursion.

The lower levels of recursion, which handle the default workspace semantics, involve 3 basic steps.

1. "Find-fewest-spanning-terms." Find the way to perform the recursive subdivide of a group of terms supplied to it. A group of terms spans a set of columns in the workspace. The algorithm finds a point between two of these columns that has the fewest number of terms spanning it. It splits the group of terms into 3 subgroups — Group 1 contains all terms that are to the left of the point, Group 2 contains all terms to the right; and Group 3 contains any terms that span the point.

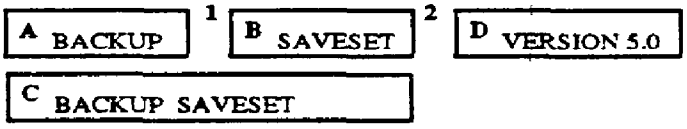
2. Groups 1 and 2 together constitute a "compatible" interpretation of the query — i.e. the results of processing the default algorithm against Group 1 can be ANDed against the results of processing the algorithm against Group 2. If Group 3 is empty, this is all that needs to be done. Otherwise, Group 3 constitutes an "incompatible" interpretation: an alternative interpretation is constructed with Group 3 (see next item), and the results of this are ORed with the AND of Groups 1 and 2.
3. "Construct-alternate-interpretation." A new group of tiles is constructed by augmenting the spanning tiles in Group 3 with some of the tiles in Groups 1 and 2. The tiles that are included from Groups 1 and 2 are any that do not overlap with at least one of the tiles in Group 3. This creates a new group with potential term overlaps in it, but later recursions will resolve these overlaps correctly. Once a group is generated in this way, it is processed recursively.

In this specific implementation, there are three degenerate cases which constitute the stopping points of the recursion.

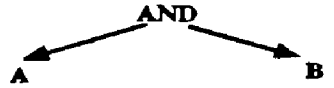
1. An empty column, which returns nothing (we ignore empty columns).
2. A single term — this is returned as a leaf node.
3. A group of multiple terms, which all span the same columns — these are ORed together.

After the tree is generated, nested Boolean AND nodes and nested Boolean OR nodes are merged, to reduce the number of Boolean operations on article sets further.

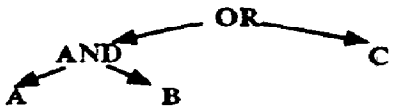
Example



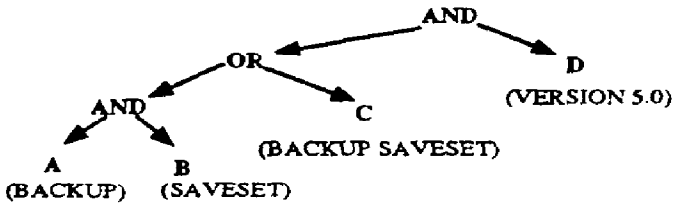
The starting group of terms is [A B C D]. "Find-fewest-spanning-terms" finds point 2 has no overlapping terms, and splits the group into Group 1, [A B C], and Group 2, [D]. Group 1 is re-analyzed, and point 1 is found as having 1 overlapping term -- we get Subgroup 1 as [A], Subgroup 2 as [B], and Subgroup 3 as [C]. Subgroups 1 and 2 are the second degenerate case — the terms are returned as nodes. Subgroup 1 and Subgroup 2 are mutually compatible, and are ANDed together



"Construct-alternate-interpretation" finds no terms in Subgroups 1 and 2 that can be included with Subgroup 3, so C is returned as the degenerate tree for the alternate interpretation, and this is ORed with the prior AND node, which gives



Finally, the results of this (the original Group 1) are compatible with Group 2, and they are ANDed together.



9. User Feedback

In a preliminary study of the effectiveness of the proposed interface, we observed a dozen users, already familiar with the STARS system, as they formulated queries using the AI-STARS interface. Users were able to manipulate the Query Reformulation Window appropriately after several minutes of explanation and experimentation. Their feedback indicated that "opening up the black box" simplified the task of query reformulation and made them more confident in the results of the search. For most realistic queries, the visual displays were readily interpretable with respect to their Boolean semantics. As alluded to in section 6, cases of "complex overlap" of tiles were difficult to decipher in the contrived examples we presented to the users. However, users did not encounter such cases while formulating queries on their own.

Further testing is required to generalize our findings to a large population. We are particularly interested in ease of learnability for novice STARS users, and comparisons of retrieval effectiveness with and without the reformulation window.

10. Performance

Our interface supports query reformulation via interactive iterative refinement. This mode of interaction requires quick query response times (i.e. within several seconds) to be most effective. Using a cache to speed up lexicon access, our current prototype, operating on a database of 3000 one-to-two page articles, satisfies our interactive response requirements. As we scale up to databases in the 100,000 to 500,000 article range (the size of the existing STARS database), we expect the major cost to come from performing I/O's on the inverted indices. For this reason, we have avoided including some of the more expensive operations sometimes available in information retrieval systems, such as term proximity constraints, word truncation, and wildcarding. Our hope is that the indexing of known contiguous phrases, morphological analysis, and the compound terms window make these search term operations largely unnecessary.

11. Related Work

A number of researchers have explored the use of graphics interfaces for information retrieval. Fischer and Nieper-Lemke [FISCHER89] support browsing and query

formulation with a visually displayed concept hierarchy. Croft and Thompson [CROFT86] allow users to graphically navigate a knowledge base consisting of a semantic network of documents, terms, and concepts, enabling users to discover and indicate relevant items as they browse. Spenke and Beilken [SPENKE89] generalize Zloof's [ZLOOF77] Query-by-Example paradigm for database retrieval with a spreadsheet based interface for interactive logic programming. McAlpine and Ingwersen [MCALPINE89] show how a graphical forms-based interface can be applied over a wide range of information retrieval tasks in the context of a "knowledge worker support system."

Our approach to the graphical representation of natural language input has also been influenced by work in "chart parsing" [KAY80], for which a two-dimensional graphical layout can be exploited to display constituent structure.

12. Conclusions

Lynch [LYNCH87] has argued that the introduction of heuristics into information retrieval systems has an associated risk, that the "portion of the user community who understands how the information retrieval system really works and how to exploit the full power of the retrieval system thus will grow smaller and smaller." Corroborating this prediction, our analysis of users' problems with the STARS full-text retrieval system revealed that the benefits of natural language queries were somewhat undermined by the necessity of hidden query reformulation.

This paper has described an approach to opening up the "black box" that characterizes many information retrieval systems accessed via natural language query. The resulting interface is a hybrid that, we believe, reaps the benefits of both natural language queries and explicit Boolean query manipulation, while minimizing the drawbacks that each approach has on its own.

Informal off-line testing by Customer Support Specialists has tended to validate this hybrid approach. Users appear to learn the visual semantics quickly and enjoy the ease with which they can experiment with the form of a query. Thus, while we have not yet had the opportunity to test our prototype on a wide number of users in a realistic setting, we are optimistic about the effectiveness of this solution for our target user community.

We are also evaluating potential shortcomings of the current approach.

- The display method is best suited for relatively short queries, since the visual representation becomes less perspicuous as the number of tiles in the configuration increases. While this limits the size of practical queries, our experience suggests that most actual queries fall within the range accommodated by this approach.
- The existence of certain complex configurations containing partially overlapping tiles (mentioned above) may make it hard for users to map some configurations to their Boolean equivalents. Again, our experience to date suggests that truly complex configurations are rarely produced in real queries, and that the interpretations of simple cases of partial overlap are easily learned by experience.
- The current system is based on mapping to an exact match Boolean expression. While the ease of manipulating this expression removes some of the drawbacks of

exact match retrieval, we have not yet considered how this approach might be adapted to an extended Boolean [SALTON83] or probabilistic [SALTON86] semantics.

In addition to such issues, future research will investigate extensions to the current set of workspace operations, the integration of structured data into a query via the workspace, and integration with other on-line retrieval aids.

As we build more intelligence into the system, enabling more system-initiated query reformulation, it will be interesting to see how well our model of user/system cooperation through the visual workspace scales up. It is our belief that the ability to display the results of system-initiated reformulation in a manner comprehensible to and easily correctable by a user will be a major factor in user acceptance of and confidence in more "intelligent" retrieval systems of the future.

References

[BATES86] Bates, Marcia J., Terminological Assistance for the On-line Subject Searcher, Proceedings of 2nd Conference on Computer Interfaces for Information Retrieval, 1986.

[BATES79] Bates, Marcia J., Information Search Tactics, Journal of the American Society for Information Science, July 1979.

[BOGURAEV82] Boguraev, B.K. and K. Sparck Jones, A Natural Language Analyser for Database Access, Information Technology: Research and Development, 1982,1,23-39.

[CROFT86] Croft, W. B. and R. H. Thompson, I3R: A New Approach to the Design of Document Retrieval Systems, COINS Technical Report 87-58, 1986.

[DEBILI88] Debili, Fathi, Christian Fluhr, and Pierre Radasoa, About Reformulation in Full-Text IRS, Proceedings of RIAO, 1988, 343-360.

[DILLON83] Dillon, M. and A. S. Gray, FASIT: A Fully Automatic Syntactically Based Indexing System, 1983, Journal of the American Society for Information Science, 34(2):99-108.

[FISCHER89] Fischer, Gerhard and Helga Nieper-Lemke, HELGON: Extending the Retrieval by Reformulation Paradigm, Proceedings of CHI, 1989, 357-362.

[FREI83] Frei, H. P. and J. F. Jauslin, Graphical Presentation of Information and Services: a User-Oriented Interface, Information Technology: Research and Development, 1983, 2(23-42).

[KAY80] Kay, M., Algorithm Schemata and Data Structures in Syntactic Processing, Xerox Palo Alto Research Center, Tech Report no. CSL-80-12, 1980.

[LANCEL88] Lancel, J. M. and N. Simonin, TEX-NAT: A Tool for Indexing and Information Retrieval, Proceedings of RIAO, 1988, 369-377.

[LYNCH87] Lynch, Clifford A., The Use of Heuristics in User Interfaces for Online Information Retrieval Systems, Proceedings of the American Society for Information Science, 1987, 148-151.

[MCALPINE89] McAlpine, Gordon and Peter Ingwersen, Integrated Information Retrieval in a Knowledge Worker Support System, Proceedings of the Twelfth Annual In-

ternational ACM SIGIR Conference on Research and Development in Information Retrieval, 1989.

[MCMATH89] McMath, Charles F., Robert S. Tamaru, and Roy Rada, A Graphical Thesaurus-Based Information Retrieval System, International Journal of Man-Machine Studies, 1989, 31, 121-147.

[RAU88] Rau, Lisa F., Conceptual Information Extraction and Retrieval from Natural Language Input, Proceedings of RIAO, 1988, 424-437.

[SALTON89] Salton, Gerard, Automatic Text Processing: the Transformation Analysis, and Retrieval of Information by Computer, Addison-Wesley, 1989.

[SALTON86] Salton, Gerard, Another Look at Automatic Text Retrieval Systems, Communications of the ACM, 1986, 29, 648-656.

[SALTON83] Salton, Gerard, Edward A. Fox, and Harry Wu, Extended Boolean Information Retrieval, Communications of the ACM, 1983, 26, 1022-1036.

[SALTON75] Salton, G., C. S. Yang, and C. T. Yu, A Theory of Term Importance in Automatic Text Analysis, Journal of the American Society for Information Science, 1975, 26(1):33-44.

[SCHWARZ88] Schwarz, Christoph, The TINA Project: Text Content Analysis at the Corporate Research Laboratories at Siemens, Proceedings of RIAO, 1988, 361-368.

[SPENKE89] Spenke, Michael and Christian Beilken, A Spreadsheet Interface for Logic Programming, Proceeding of CHI, 1989, 75-80.

[THOMPSON89] Thompson, R. H. and W. B. Croft, Support for Browsing in an Intelligent Text Retrieval System, International Journal of Man-Machine Studies 1989, 30, 639-668.

[ZLOOF77] Zloof, Moshé M., Query-by-Example: a Data Base Language, IBM System Journal 16(4), 1977, 324-343.