Improving User Performance on Boolean Queries

John F. Pane
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213 USA
+1 412 268 8078
pane+chi2000@cs.cmu.edu

Brad A. Myers
Human Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA
+1 412 268 5150
bam+@cs.cmu.edu

ABSTRACT

The accurate formulation of boolean expressions is a notorious problem in programming languages as well as database and web query tools. Users have demonstrated great difficulty with the common textual method for specifying these queries, which uses the boolean operators *AND*, *OR*, and *NOT*, partly because these words are used inconsistently in natural languages. This paper proposes a tabular boolean query language that avoids the need to use named operators, provides a concrete distinction between conjunction and disjunction, and makes grouping more explicit. A study comparing this tabular language with textual boolean expressions found that untrained users perform better when they express their queries in the tabular language, and about equally well when interpreting queries written in either language. We conclude that systems can benefit by adopting a tabular notation for query formulation.

Keywords

End-user programming, psychology of programming, user studies, information retrieval.

INTRODUCTION

We are using a human-computer interaction approach to the design of a primarily textual programming system for children. One of the challenges of this effort is to craft the features of the system to address the problems researchers have observed in prior systems. While this may be straightforward in some cases, it is quite difficult in others.

The accurate specification of boolean expressions is a notorious problem area in programming languages as well as common database retrieval tasks such as searching library catalogs or the web. Researchers have observed that the common uses of the words *AND*, *OR*, and *NOT* in natural language lead to errors in the use of these words to name the boolean operators in queries. For example, it is common for users to substitute *AND* for *OR* [2], and the intended scope of the *NOT* operator is ambiguous [3]. The difficulties with boolean queries are intensified when several operators must be combined to form the query, and they persist even when parenthesis are used to clarify grouping [2]. Textual languages that attempt to circumvent these issues by using different vocabulary and query structure do not seem to be effective [4]. Because no universally better alternatives have been discovered, most programming languages continue to rely on textual boolean expressions, while many search engines have turned to less expressive query languages (for example, the plus and minus unary operators for inclusion and exclusion). *Newsweek* reports that even with these simplifications, most web users are dissatisfied with search engines, and less than 6% manage to use these operators in their searches [5].

Appeared in *CHI 2000 Extended Abstracts: Conference on Human Factors in Computing Systems*, G. Szwillus and T. Turner, Eds. The Hague, Netherlands: ACM Press, April 1-6 2000, pp. 269-270.

Several researchers have developed graphical interfaces to the boolean query task. For example, truth tables, Venn diagrams, tiles in a two-dimensional grid, electrical circuits, and water flowing through filters have all been used with varying success. However, many of these interfaces are limited to very simple queries, and they use a large amount of screen space. They would be difficult to incorporate into a textual programming language where the boolean query must be written and viewed in the context of the surrounding program.

PROPOSED TABULAR QUERY LANGUAGE

We propose an alternative query language that uses a tabular layout. Since our new programming language will represent data on *cards* containing slots with values, we designed *match forms* to use cards as well. Figure 1 shows a sample query. Criteria are placed into slots, one term per slot. As many slots as necessary can be used, and they implicitly form a conjunction. Negation is specified by prefacing a term with the *NOT* operator. Disjunction is specified by placing one or more additional match forms adjacent to the first.

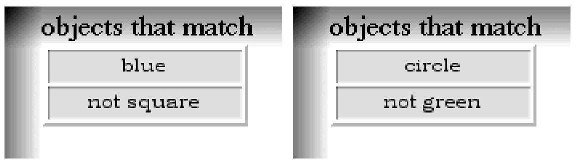

Figure 1: Match forms expressing the query: (blue and not square) or (circle and not green)

Match forms avoid using the troublesome *AND* and *OR* keywords by making those boolean operators implicit in the structure of the query. This two-dimensional layout is similar to the grid of tiles described by Anick et al. [1], with some additional cues to help users understand which operator uses each dimension. The scope of the NOT operator is made explicit by confining it to a single term. This proposed query language can express arbitrarily complex queries, although it requires some queries to be formulated in a less concise way than full-fledged boolean expressions would permit. Our proposed language will also provide a way to negate an entire form, but that feature was not used in the study described here.

We hypothesized that users would perform more accurately in writing and reading queries that use match forms than they would on equivalent queries that use textual boolean expressions.

THE STUDY

The study used a grid of nine colored shapes, where a subset of the shapes could be marked. The participants were given 13 problems: a group of 5 *code generation* problems, where some shapes were already marked and they had to formulate a query to select them; followed by a group of 8 *code interpretation* problems, where they were shown a query and had to mark the shapes selected by the query. They solved each of these problems twice, once using a purely *textual language*, and once using *match forms*. Each group of problems was preceded by instructions which we constructed to be as similar to each other as possible. We varied the sequence in which the participants solved the problems to prevent presentation order from impacting the results. They were instructed to be as accurate as possible and were told that there was no time limit.

33 volunteers participated in the study, 13 children (ages 10-14), and 20 adults (ages 18-46). 14 of the participants were male and 19 were female. All but two were native speakers of English. 7 participants reported that they had written computer programs (4 adults, 3 children).

RESULTS

The hypothesis was evaluated by comparing each participant's performance between the two conditions: textual expressions and match forms. Statistical significance was evaluated with a non-parametric sign test, with $p<.05$ as the threshold for significance. No significant differences were detected between children and adults, between males and females, or between programmers and non-programmers, so the following results are aggregated across all of the participants.

On the code interpretation problems, participants answered 71% of the problems correctly when match forms were used, and 74% correctly when textual queries were used. This difference is not significant. However, on the code generation problems, the participants answered 94% of the problems correctly when using match forms, and only 85% correctly when using textual queries. This difference is significant ($p<.0001$).

DISCUSSION

On code interpretation tasks, the participants performed about as well with match forms as they did with text. However, on code generation tasks, the participants performed significantly better with the match forms than they did with text. In addition, we informally observed that as the queries became more complex the differences in favor of the match forms increased. Although the match forms were not superior for code interpretation, they were not detrimental either. The strong positive impact for generation probably outweighs any ambivalence about the lack of an effect for interpretation.

The strong effect of match forms over text came with very little training. It is unlikely that the participants ever used an equivalent tabular query language before, and they only viewed a brief instruction page with a few examples before beginning to solve the problems. While the instructions for the textual problems were similarly brief, the participants came to the problems with knowledge from a lifetime using the words *AND*, *OR*, and *NOT* in English, which may have hindered them as much as it helped them.

CONCLUSION

The results of this study suggest that the usability of web search engines, database query tools, and programming languages can be improved, particularly for untrained users, by replacing textual boolean expressions with tabular forms for expressing queries. The match forms proposed here circumvent some serious problems that have been observed with textual boolean expressions, yet are compact enough to use in situations where screen space is limited.

ACKNOWLEDGMENTS

We would like to thank Albert Corbett, Bernita Myers, and Barbara Pane for their contributions to this research, as well as all of the anonymous participants. This research is funded in part by the National Science Foundation under Grant No. IRI-9900452. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

REFERENCES

1. Anick, P. G., Brennan, J. D., Flynn, R. A., Hanssen, D. R., Alvey, B., and Robbins, J. M. A Direct Manipulation Interface for Boolean Information Retrieval via Natural Language Query. In *Proceedings of the Thirteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. User Interfaces*. (Brussels, Belgium, September 5-7 1990), 135-150.
2. Greene, S. L., Devlin, S. J., Cannata, P. E., and Gomez, L. M. No IFs, ANDs, or ORs: A Study of Database Querying. *International Journal of Man-Machine Studies, 32,* 3 (1990), 303-326.
3. McQuire, A. and Eastman, C. M. Ambiguity of Negation in Natural Language Queries. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Posters: Abstracts.* (1995), 373.
4. Pane, J. F. and Myers, B. A. Natural and Accurate Ways to Specify the Selection of Objects from a Group. *submitted for publication* (2000), http://www.cs.cmu.edu/~pane/study3.html.
5. Tanaka, J., “The Perfect Search,” in *Newsweek*, 134, 13, (September 27 1999), 71.