Kleinberg, I. (1993). Programming Knowledge: On Indexing Software for Reuse and not Indexing Documentation at All. 4th ASIS SIG/CR Classification Research Workshop, 139-146. doi:10.7152/acro.v4i1.12616

Programming Knowledge: On Indexing Software for Reuse and not Indexing Documentation at All

Ira Kleinberg
School of Communication, Information and Library Studies
Rutgers. The State University of New Jersey
4 Huntington Street
New Brunswick, NJ 08903
kleinberg@zodiac.rutgers.edu

This paper notes something of a paradox: Computer software reuse systems both intellectual and automated are making use of humans and documentation at a time when the role of both humans and documentation is being called into question by the computer industry itself. The challenge facing classification researchers and practitioners is to show how reuse systems of all types depend on natural-language representations of the texts they contain, and to share the methods that classification research and practice have developed to create those representations.

The media has seemingly been filled with stories on the subject of reuse in recent months, everything from the reuse of television shows made available on demand thanks to a virtually unlimited number of channels (Dibbell, 1992, December 22), to the reuse of music in the form of CDs created on the spot in stores without inventories (Lohr, 1993, May 12), to the reuse of text in books tailored to individual specifications and "printed" on an as-needed basis (Cox, 1993, June 1). At the heart of all these reuse systems is the idea that information (the TV shows, the music, the text) will be organized into some kind of "library," with everything that word implies; stored in digital form; retrieved electronically using some kind of search interface; and then delivered to users via phone, cable, satellite, or other means.

The issues involved in reuse may no doubt sound like the kinds of issues in which classification researchers and practitioners — defined here as those with a theoretical or practical interest in the cataloging and indexing of "text," used here in the broadest possible sense of the word--should be involved. But are they? The purpose of this paper is to answer that question more fully and from one specific context, that of the computer industry.

Drawing on examples from the computer science literature and the author's own experience in the technical documentation field, this paper notes something of a paradox: Computer software reuse systems both intellectual and automated are making use of humans and documentation at a time when the role of both humans and documentation is being called into question by the computer industry itself. Computer software reuse systems organize and represent individual program components so that they can be stored, retrieved, adapted, and essentially reused at some future point in time. Intellectual reuse systems depend on the cataloging and classification efforts of humans. Automated reuse systems depend on the natural-language documentation associated with the program components. Humans can participate in the process as programmers, writers, catalogers, and indexers. Documentation can consist of manual pages and comments.

This paper suggests that the challenge facing classification researchers and practitioners with regard to reuse systems of all types is not to "prove" that systems based on intellectual retrieval

ISSN: 2324-9773

Kleinberg, I. (1993). Programming Knowledge: On Indexing Software for Reuse and not Indexing Documentation at All. 4th ASIS SIG/CR Classification Research Workshop, 139-146. doi:10.7152/acro.v4i1.12616

methods are "better" than those based on automated ones. Rather, the challenge facing researchers and practitioners is to show how reuse systems both intellectual and automated depend on natural-language representations of the texts they contain, and to share the methods that classification research and practice have developed to create those representations.

ON INDEXING SOFTWARE FOR REUSE...

"How would we index the sample code on our CD-ROM for developers?" an employee of a major computer hardware manufacturer asked the author late last year. The sample code consisted of program components created in house to show off the company's hardware to best advantage. The CD-ROM was comprised of some 650 Megabytes of technical information which, in addition to the sample code, included manuals, bug reports, demonstration programs, and magazines. The developers were outside programmers who made the software that ran on the company's hardware.

The employee's question just happened to coincide with a session on the classification and retrieval of computer software presented at the American Society for Information Science's 55th annual meeting (American Society for Information Science, 1992). Papers suggested by that session offered a number of possible solutions to the software reuse problem, including intellectual solutions (Prieto-Diaz, 1991), automated solutions (Maarek, Berry, & Kaiser, 1991), and solutions that combined the two (Frakes & Pole, 1992).

An Intellectual Software Reuse System

Prieto-Diaz (1991), for example, reported on two implementations of an intellectual system for software reuse. This system uses faceted classification to describe the components in the reuse library — that is, classification based on different aspects of the component being classified (such as type of component or intended method of use). The system also depends to a large extent on the efforts of a librarian, management, and the users themselves to keep it going.

Prieto-Diaz reported that he chose intellectually based faceted classification, rather than automated free-text retrieval, because it appeared that most of the characteristics of source code (such as lack of free text or ambiguity in the "meaning" of the code itself) made it unsuitable for effective automated retrieval. In testing a version of one of his prototypes using faceted classification against another version using no classification, he found a four-fold increase in recall and precision. Prieto-Diaz also reported a reuse factor of 14 percent after 1 year for this particular system, and was predicting a reuse factor of 50 percent after 5 years.

An Automated Software Reuse System

Maarek, Berry, and Kaiser (1991), in contrast, reported on an automated system for software reuse--in this case, a system that extracts natural-language documentation associated with or included in programs and then uses lexical affinities (correlation between the appearance of two phrases in a body of text) and hierarchical clustering (a clustering technique based on those lexical affinities

ISSN: 2324-9773

Kleinberg, I. (1993). Programming Knowledge: On Indexing Software for Reuse and not Indexing Documentation at All. 4th ASIS SIG/CR Classification Research Workshop, 139-146. doi:10.7152/acro.v4i1.12616

that facilitates browsing) to allow users to both query and browse the system in search of potentially useful components.

Maarek, Berry, and Kaiser reported that they chose automated retrieval primarily for economic reasons, and focused on the documentation associated with or included in programs because of the lack of conceptual information in the programs themselves. The researchers compared their system using an "information retrieval" approach (based on free-text indexing) to another system using what they termed the "artificial intelligence" approach (based on domain analysis and information manually input into a knowledge base), and found that it performed similarly in terms of recall but 15 percent better in terms of precision. However, the researchers were quick to point out that their results were not, as yet, statistically significant.

A Combined Intellectual/Automated Software Reuse System

Frakes and Pole (1992), taking a different approach, reported on an empirical study of four representation methods for reusable software components. These methods included enumerative classification (an intellectual representation method based on precoordinated hierarchical classes), faceted classification (the intellectual representation method discussed above), attribute-value classification (an intellectual representation method similar to faceted classification), and keyword indexing (the automated representation method discussed above). Searching effectiveness (recall, precision, overlap), user preference, and search time were among the things measured in the study.

There were a number of results of interest in Frakes and Pole's study. Among them: there were no significant differences in recall and precision between the four representation methods, with each method performing only moderately well in terms of search effectiveness; each method found different items, with overlaps between pairs of documents averaging about 80 percent; no one representation method was preferred by users above any other; there were significant differences in search times between the representation methods, with enumerative classification performing an average of 60 percent faster than keyword indexing.

A Few Conclusions

Note that these software reuse projects, although different in the manner in which they went about facilitating reuse, shared many similarities. Among them:

- They made use of methods that came out of classification research and practice--methods both intellectual and automated.
- They made use of humanly created representations of software code for retrieval purposes, rather than the code itself--including facet analyses, comments, and documentation.
- They were geared toward meeting the needs of users--needs defined through user studies and refined through user feedback.

ISSN: 2324-9773

Kleinberg, I. (1993). Programming Knowledge: On Indexing Software for Reuse and not Indexing Documentation at All. 4th ASIS SIG/CR Classification Research Workshop, 139-146. doi:10.7152/acro.v4i1.12616

- They were not created to be sold as "product"--they were created to serve in-house needs.

- They required a long-term commitment from the highest levels of the organization — a commitment of personnel, time, and money. That is not to say, however, that the organization did not expect a substantial return on its investment.

Note also that the results from the one comparative study (Frakes & Pole, 1992) echo those found in numerous places in the information science literature (for example, Katzer, McGill, Tessier, Frakes, & DasGupta, 1982; Salton, 1989; Saracevic, 1991). Those results suggest that different retrieval methods perform quite similarly in terms of precision and recall, but quite differently in terms of the actual documents retrieved and the amount of time it takes to retrieve them.

...AND NOT INDEXING DOCUMENTATION AT ALL

How do these examples drawn from the computer science literature compare to the author's own experience in the technical documentation field?

A Sample Code Index

Consider again the case of the computer hardware manufacturer interested in indexing its sample code: Work on the project was halted before it ever really began. As the employee in charge put it, "Human indexing is just taking too long and costing too much money. And we think our users should be able to find 60-80 percent of what they're looking for using Boolean retrieval."

These statements were made without the benefit of any kind of cost-benefit analysis or user testing. And users presumably continue to access the sample code the same way they always have: by searching on or visually inspecting the names of hundreds, perhaps thousands, of files and subdirectories.

A Reusable Art Library

To use another example, a computer book publishing company was concerned about its ability to keep track of the art (the photos, line drawings, and illustrations) that appeared in its books. This art existed in both traditional (on boards) and electronic (on disk) form, numbered some 20,000 pieces, and was stored throughout the company with minimal organization.

In theory, a piece of art created for one book could be reused in another; in practice, it first had to be found. Although it was estimated that 50 percent of the company's art (10,000 pieces) was potentially reusable, it also was estimated that 25 percent of that art (5000 pieces) would never be found and would have to be re-created. Because of the company's desire to both cut costs in the

ISSN: 2324-9773

Kleinberg, I. (1993). Programming Knowledge: On Indexing Software for Reuse and not Indexing Documentation at All. 4th ASIS SIG/CR Classification Research Workshop, 139-146. doi:10.7152/acro.v4i1.12616

short term and enter the custom publishing market in the long term, this was seen as a potential problem.

The solution proposed--creating a library so that all the company's art could be stored in one place, hiring a librarian so that one person could take charge of that art, and organizing a catalog so that anyone interested could search for and retrieve a necessary piece of art — was not received with great enthusiasm. One reason was that it was an intellectual, rather than an automated, solution to the problem at hand. Another reason was that it was a solution that would cost the company as much money to implement and run as it would save. In the short term, there was no money for the company to implement the solution; in the long term, there was no profit for the company to make. To date, plans for the company's art, and its custom products, remain up in the air.

"Embedded" Indexes

In yet another example, a computer software firm was interested in saving time (and money) on the indexes it was producing for its documentation. Over the years, the company had gone from providing no indexes to its documentation, to providing author-created indexes, to providing professional indexer-created indexes. However, because of schedule (and budget) constraints, the company decided to return to author-created indexes, but this time with the help of the indexing module of its new document-processing software.

Now, authors indexed as they wrote, "embedding" index terms in the document itself, and compiling the finished index when the document was finished. The only problem was, when a mistake was discovered in the "finished" index, the indexing module required the author to go back and correct the term embedded in the "finished" document. This process frequently would be repeated several times.

The company was told that, based on the experience of professional indexers, these so-called "embedded" indexes usually took twice as long, cost twice as much, and were of half the quality of traditional indexes compiled in "stand-alone" fashion. In addition, the company was told that more than half the job of a professional indexer was editing the in-progress and finished index, something that these indexing modules did not easily allow. Finally, the company was told that when it came time for revision--especially extensive revision--it was frequently easier to scrap the existing index and start from scratch.

As a result of this advice and its experience with author-created embedded indexes, the company is currently experimenting with a new policy: "not indexing" some of its documentation. So far, the feedback has been quite positive. As one employee reported, "No one's complained."

Obvious Products

Finally, in an example drawn from the technical documentation literature but quite typical of the feeling in the field itself, Horton (1993) suggests that technical documentation is doomed and that technical writers would be well advised to consider future careers as product designers. "The right

ISSN: 2324-9773

Kleinberg, I. (1993). Programming Knowledge: On Indexing Software for Reuse and not Indexing Documentation at All. 4th ASIS SIG/CR Classification Research Workshop, 139-146. doi:10.7152/acro.v4i1.12616

way to eliminate paper documentation is to make the product so nearly obvious that it needs little documentation and then to provide that documentation electronically as an integral part of the product" (p. 27).

A Few More Conclusions

These examples drawn from the technical documentation field would appear to stand in opposition to those drawn from the computer science literature. First, the examples suggest a move away from the act of representing a text, particularly a non-natural-language text, in anything other than its original form. These other representations could include indexes, catalog records, or documentation. Next, the examples suggest a move away from the use of the people traditionally thought to be able to create such representations. These people could include indexers, librarians, or writers. Finally, the examples hint at the return of publishing in general to its roots as a cottage industry. The availability of and belief in new technology, along with a corresponding desire to save time and cut costs, could be seen as helping to foster this return. (For more on the foundations of publishing, see Coser, Kadushin, & Powell, 1982. For more on the foundations of librarianship, see Shera, 1972. For more on the foundations of information science, see Borko, 1968.)

PROGRAMMING KNOWLEDGE: THE CHALLENGE FACING CLASSIFICATION RESEARCHERS AND PRACTITIONERS

The challenge facing classification researchers and practitioners with regard to reuse systems of all types is not to "prove" that systems based on intellectual retrieval methods, methods using indexers or librarians or writers, are "better" than those based on automated ones. Rather, the challenge facing researchers and practitioners is to show how reuse systems both intellectual and automated depend on natural-language representations of the texts they contain, representations such as indexes or catalog records or documentation, and to share the methods that have been developed to create those representations, methods such as indexing or cataloging or technical writing.

REFERENCES

American Society for Information Science. (1992). SIG/CR — Classification and retrieval for software reuse. In D. Shaw (Ed.), *ASIS '92: Proceedings of the 55th ASIS annual meeting* (Vol. 29, pp. 316-317). Medford, NJ: Learned Information.

Borko, H. (1968). Information science: What is it? *American Documentation*, 19(1), 3-5.

Coser, L. A., Kadushin, C., & Powell, W. W. (1982). *Books: The culture and commerce of publishing*. New York: Basic Books.

Cox, M. (1993, June 1). Technology threatens to shatter the world of college textbooks. *Wall Street Journal*, pp. A1, A6.

Dibbell, J. (1992, December 22). It's the end of TV as we know it (and I feel wired). *Village Voice*, pp. 55, 60.

Frakes, W. B., & Pole, T. (1992). An empirical study of representation methods for reusable software components. Manuscript submitted for publication.

ISSN: 2324-9773

Kleinberg, I. (1993). Programming Knowledge: On Indexing Software for Reuse and not Indexing Documentation at All. 4th ASIS SIG/CR Classification Research Workshop, 139-146. doi:10.7152/acro.v4i1.12616

Horton, W. (1993). Let's do away with manuals...before they do away with us. *Technical Communication*, 40(1), 26-34.

Katzer, J., McGill, M. J., Tessier, J. A., Frakes, W., & DasGupta, P. (1982). A study of the overlap among document representations. *Information Technology: Research and Development, 2*, 261-274.

Lohr, S. (1993, May 12). Record store of near future: Computers replace the racks. *New York Times*, pp. A1, D13.

Maarek, Y. S., Berry, D. M., & Kaiser, G. E. (1991). An information retrieval approach for automatically constructing software libraries. *IEEE Transactions on Software Engineering*, 17(8), 800-813.

Prieto-Diaz, R. (1991). Implementing faceted classification for software reuse. *Communications of the ACM*, 34(5), 88-97.

Salton, G. (1989). *Automatic text processing: The transformation, analysis, and retrieval of information by computer.* Reading, MA: Addison-Wesley.

Saracevic, T. (1991). Individual differences in organizing, searching and retrieving information. In J.-M. Griffiths (Ed.), ASIS '91: *Proceedings of the 54th ASIS annual meeting* (Vol. 28, pp. 82-86). Medford, NJ: Learned Information.

Shera, J. H. (1972). *The foundations of education for librarianship*. New York: Becker and Hayes.

ISSN: 2324-9773

Kleinberg, I. (1993). Programming Knowledge: On Indexing Software for Reuse and not Indexing Documentation at All. 4th ASIS SIG/CR Classification Research Workshop, 139-146. doi:10.7152/acro.v4i1.12616

ISSN: 2324-9773