Visualizing Texts

Darrell Raymond
Department of Computer Science
University of Waterloo

August 1993

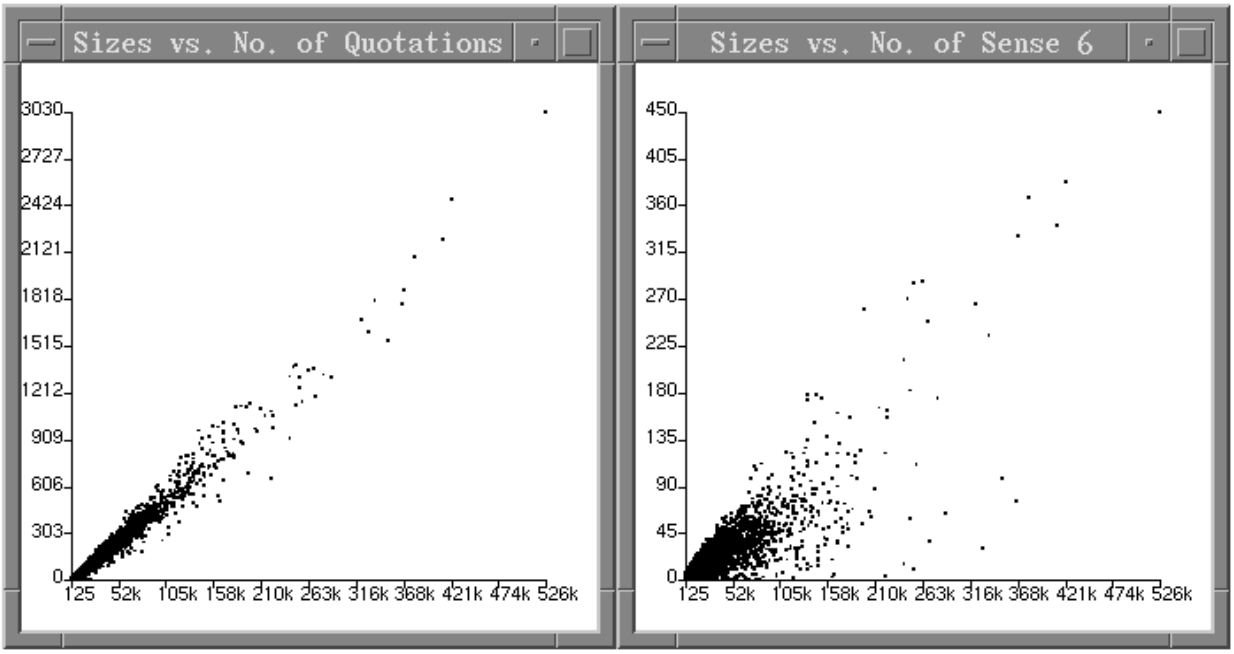
Abstract

Comprehension of the structure of text can be aided by graphical display. I describe a program that supports interactive graphical display of information about texts. The main features of the program are the use of the display as a query language, and the ability to interconnect several displays to provide a multidimensional view of the text. The formal properties of graphical querying are discussed.

1 Introduction.

Finding the line of demarcation between word and picture is an enduring philosophical problem. Many dichotomies have been proposed, including nature-convention[1], dense-discrete[2], discursive-depictive[3], sublime-beautiful[4], space-time[5], complex-salient[6], holistic-decomposable[7], hot-cold[8], and token-iterative vs. token-slotting[9]. The existence of many explanations is a tacit admission of the lack of a convincing one. It also suggests that the question itself is ill-formed. We seem unable, however, to disabuse ourselves of the notion that writing and picturing are fundamentally different forms of communication.

Scientific, mathematical, and engineering work has long operated on both sides of this distinction. While the fundamental laws of these disciplines are expressed in highly formal writing, much of their intuition is derived from pictorial representations. Consequently, there has been a longstanding desire to elevate pictures to a more formal status. In mathematics, for example, Littlewood suggested that suitably rigorous pictures be accorded the status of written proofs[10], a notion that presaged the 'proofs without words' section of *Mathematics Magazine*. More recently, several disciplines have flirted with 'computer visualization.'[11,12] This is the graphical simulation of complex phenomena, with the intent of exposing relationships that are not salient when presented in textual form. Of course, even the earliest cartographers and geometers had the same goal; what makes computerized visualizations different is that they can simulate larger and more complex phenomena than can be done manually, and that they permit interaction with the simulations.



(a) size vs. no. of quotations (b) size vs. no. of sense-6 tags

Figure 1: *OED* statistics.

In this paper I make two simple claims. The first is that visualization can be an important technique for studying texts. Making sense of words implies the ability to discern groupings; a good graphic, by definition, is an inscription in which important groupings are highly salient. My second, more novel claim is that the groupings in a visualization are as useful for asking questions about the data as they are for displaying its characteristics. Almost any plot of real-world data exhibits both a trend and some exceptions to the trend, and hence raises the natural question 'which elements of the world correspond to the trend, and which to the exceptions?' The groupings of a graphic are, in some sense, a vocabulary that can be used to query the data. Though existing visualization software naturally induces such queries, its capacity to answer them is undeveloped. Such queries also raise interesting theoretical issues. I have been attempting to make some progress in this area.

2 Visualizing the *Oxford English Dictionary.*

My interest in visualizing texts began when I had occasion to graphically illustrate some of the characteristics of the *OED.*[1] The structure of the *OED* is sufficiently detailed to warrant a book-length treatment[13], but not all its users can take the time to learn the

[1] Murray himself used a graphical depiction of the 'circle of the English language' in his Preface to the first Edition.

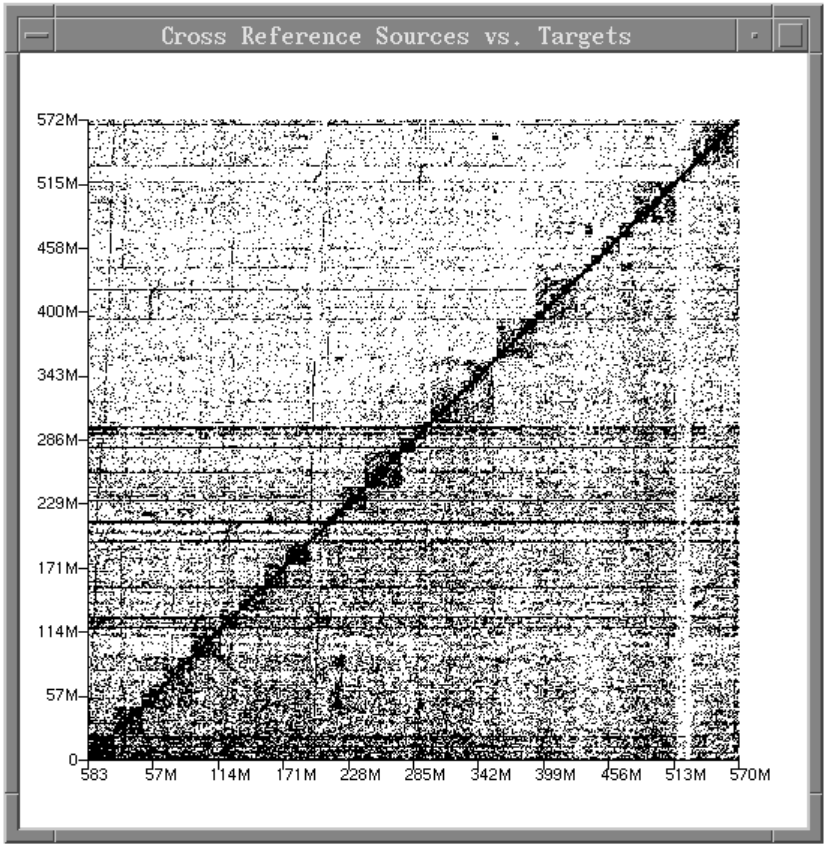
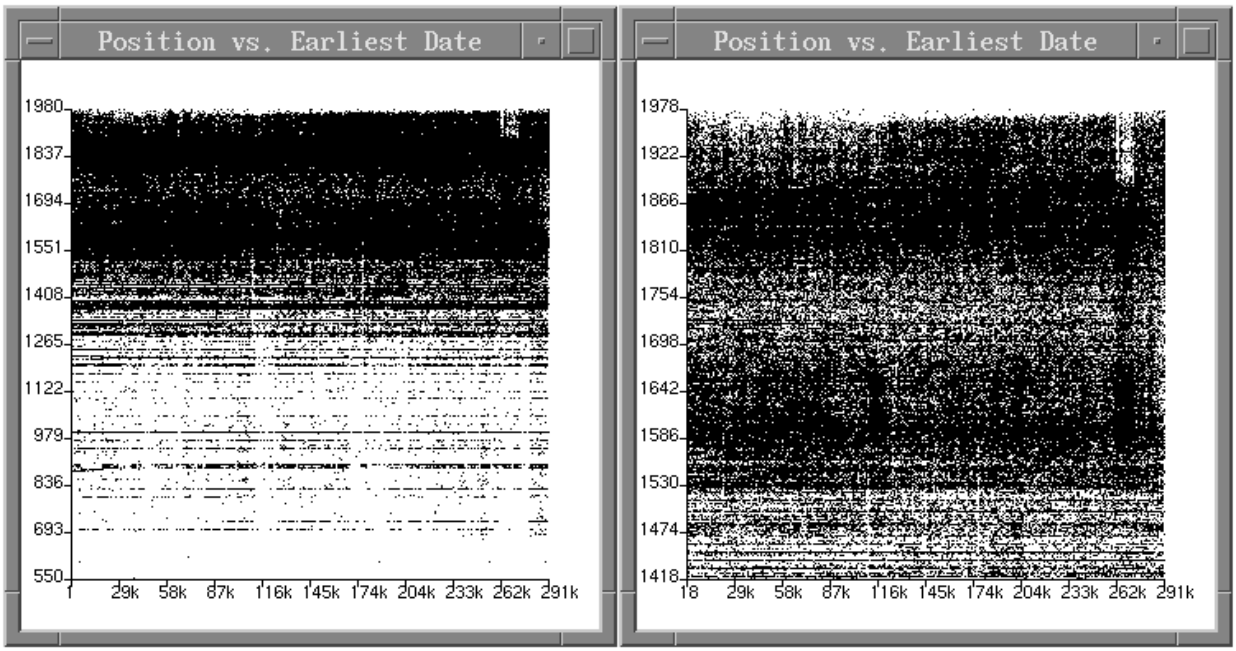


Figure 2: Cross references.

Dictionary in this way. Our goal was to effectively display some of the *OED*'s properties in order to make a convincing argument that the *OED* posed important problems for conventional hypertext systems. Two of these graphs are shown in Figure 1.

Figure 1a shows a plot of entry size versus the number of quotations in an entry; Figure 1b shows a plot of entry size versus the number of 'sense 6' tags (sense-6 tags are a rough measure of the entry's hierarchical structure). What the graphs indicate is that the number of quotations is a good predictor of the overall size of an entry (and vice versa), but that the number of sense-6 tags is not; some large entries have relatively few sense-6 tags, while other small entries have relatively many. We concluded from this and other data that simply showing hierarchical structure is a misleading way to present the entry[14].

These graphs were originally produced as static figures, using standard software for creating paper documents. Curiosity about the words associated with the outliers in the plots, however, led me to build an interactive visualizer that supports various types of manipulation and querying of plots. The visualizer is a general-purpose program, not limited to the *OED* or to these specific sets of data. Arbitrary relationships can be easily displayed, so long as they can be represented as a collection of binary instances.

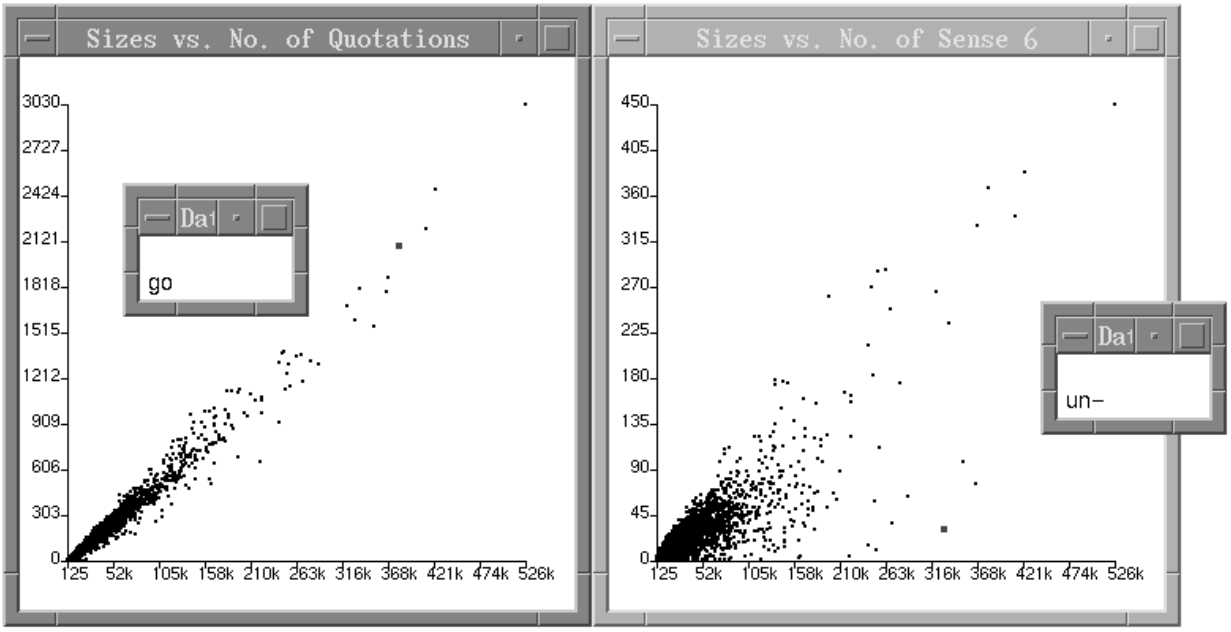


(a) (b)

Figure 3: Standard and zoom views of earliest quotes.

For example, quite a different-looking plot is shown in Figure 2, which was suggested to me by Frank Tompa. This graph illustrates the distribution of cross-references in the *OED*.

Each cross-reference is a pair of numbers: *(source, destination)*. Source locations are plotted on the horizontal axis, and destination locations are plotted on the vertical axis. The graph illustrates several interesting properties about the cross-reference structure of the *OED*. In general, darker locations indicate greater density of cross-reference targets. The dark diagonal line shows that most cross-references are to nearby locations. The diagonal line divides the plot into two triangles; the slightly greater density of the lower triangle indicates that there are more backward references than forward references. The slightly greater density of the lower half of the plot indicates that there are more references to the first half of the Dictionary than to the second half. These characteristics are likely due to the method of the Dictionary's production; it would have been easier to reference completed parts of the text than those that were yet to be done, and easier still to reference parts of the text that had been completed recently or were about to be worked on. The dark horizontal lines correspond to entries that are referenced from all parts of the text; these tend to be afixes such as **un-**, **-ous**, and **-ic**. Finally, the small rectangles of density that are found on the main diagonal correspond to parts of each letter of the alphabet. These rectangles show that there is a higher degree cross-referencing within a given letter of the alphabet than outside of it.

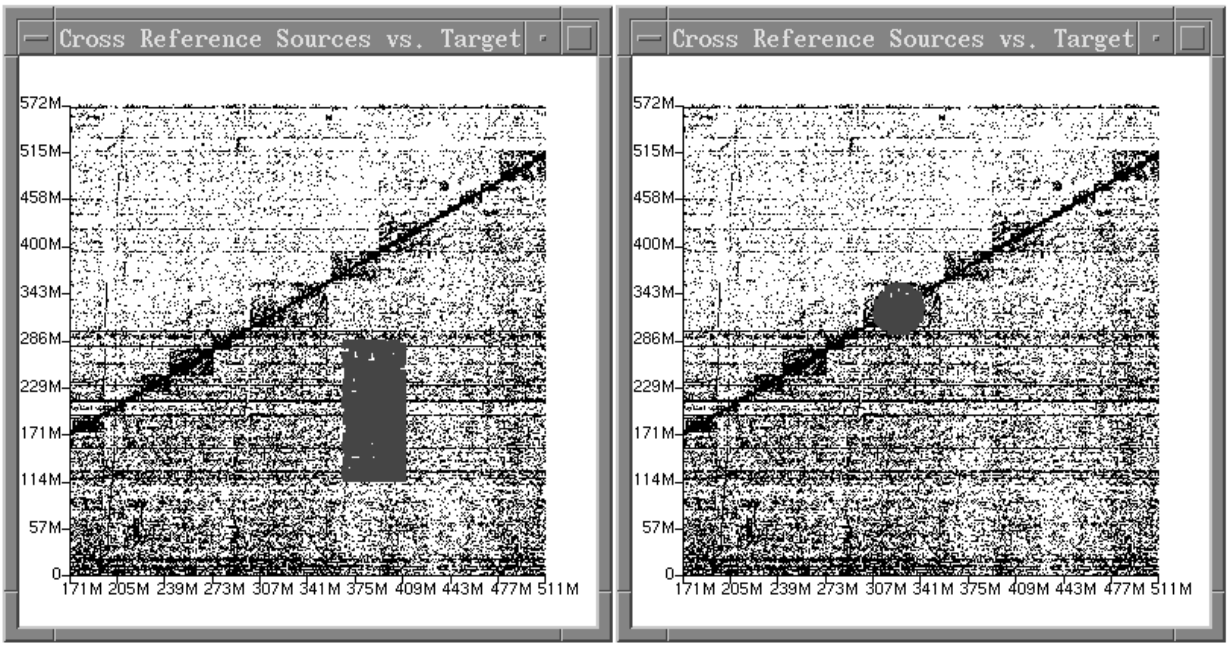


(a) (b)

Figure 4: Point identification queries.

Graphs of this sort have long been known to be useful summaries of data[15]. The effort of constructing such a graph, however, meant that one would only plot data likely to produce a worthwhile result. One benefit of computerization is in reducing the effort of plotting, and so it is possible to do 'speculative plotting' . Given a collection of statistics on a text, it is easy to arrange for each value to be plotted against the others, and to quickly scan the plots for interesting patterns.

Computerized visualizations, unlike paper ones, can be interactive. A simple kind of interaction is to change the scale and scope of the plot. Graphs can be studied in greater detail by making the whole plot larger, by stretching one of the axes, or by using the visualizer's 'zoom' function to expand a selected part of the plot. Figure 3 shows the use of the zoom function. Figure 3a shows the dates of the earliest quotes for each entry; the horizontal axis is the entry's lexicographic position in the Dictionary (the axis values indicate the ordinal number of the entry). This graph shows that many more words have earliest citations dating from after the 1550's than before. The horizontal lines result from frequently-employed dates. In some cases these are due to works that are heavily cited, such as *Blickling Homilies*. Other cases probably result from a 'stairstep' phenomena: when you don't know the exact date, you pick a round number, and the earlier the date is, the more likely you are to round it off to a larger value (hence the



(a) bounding box query (b) radial query

Figure 5: Set identification queries.

near-solid line at 1000 A.D.) The structure of the dense mass of points from about 1400 on is revealed in the zoomed plot of Figure 3b. The plot shows a regular variance that is almost sinusoidal. It would be interesting to know whether the pattern was due to the Dictionary's choice of sources, or due to some natural or historical phenomenon of productivity in English.

The visualizer also supports querying the graph. One kind of query is point identification. When the user requests point identification, a secondary window is opened, and the labels associated with individual points are displayed in this window as the cursor is swept near them. Figure 4 shows point identification in use on the graphs of Figure 1. Figure 4a shows the cursor identifying the fourth-largest entry in the *OED* (**go**). Figure 4b shows the cursor identifying one of the large entries that has relatively few sense-6 tags (**un-** (due to the method used to capture the screen image, the cursor itself is not visible. The identified entries are denoted by slightly larger squares).

It is important that the point identification feature operate efficiently, so that the user can sweep the cursor through a dense set of points and get a quick overview of the contents. By rapidly checking the points in the range of **un-**, for example, one learns that many of the large entries with a small number of sense-6 tags are either prefixes (**non-**, **iso-**, **semi-**, **self-**) or words that participate in many compounds, such as **water**, **red**, **sea**, **white**, **wind**, and **one**. A substantial part of the content of such entries is a list of words formed from the lemma.

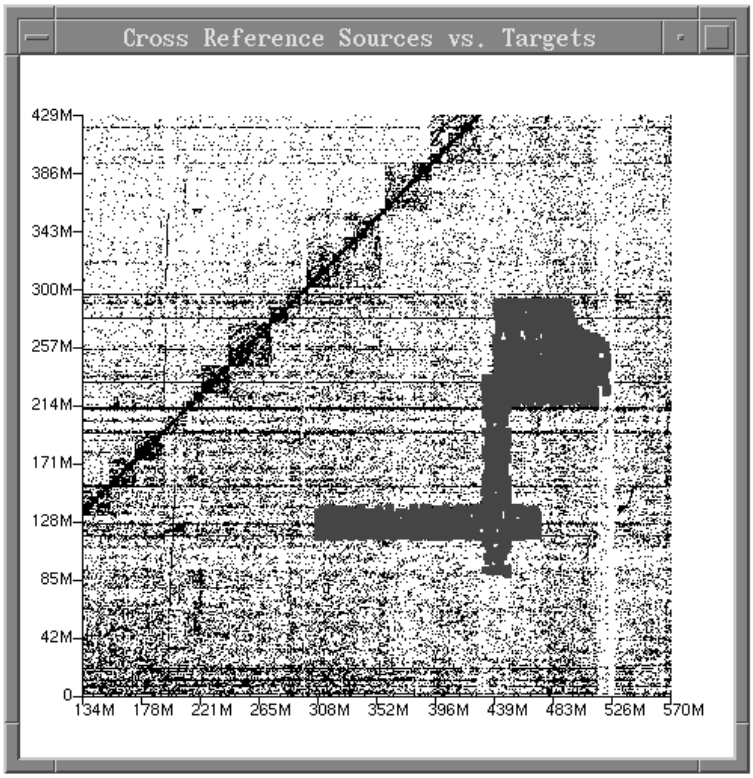


Figure 6: Boolean querying.

The visualizer also supports a range of set identification queries. One simple tool for determining a set is the bounding box. Using a rubber-band rectangle, the user specifies a rectangular box on the plot, and the points that fall within the bounds of the box become the set of interest. Figure 5a shows a plot after a bounding box query has been specified.

Another type of set identification query is the radial query. In this query, the user specifies a centre point and a radius, and all points that fall within the designated circle are returned. Figure 5b shows a plot after a radial query has been specified.

Bounding box and radial queries are essentially user interface techniques to specify vector norms[16]. The radial query corresponds to the l_2 norm, while the bounding box is very similar to the l_∞ norm. In effect, by drawing a bounding box, the user is asking for the set of points that are 'close' to some central point, where the notion of closeness is defined by the type of box (rectangular or circular).

The radial query has a natural iterative mode; one can easily imagine iteratively extending the radius to obtain an expanded set. Bounding box queries can also be handled iteratively, by simply allowing the specification of multiple boxes. The visualizer shown here allows iterative queries to be constructed from a sequence of box or radial queries. Furthermore, it allows any of the standard boolean operators—union,

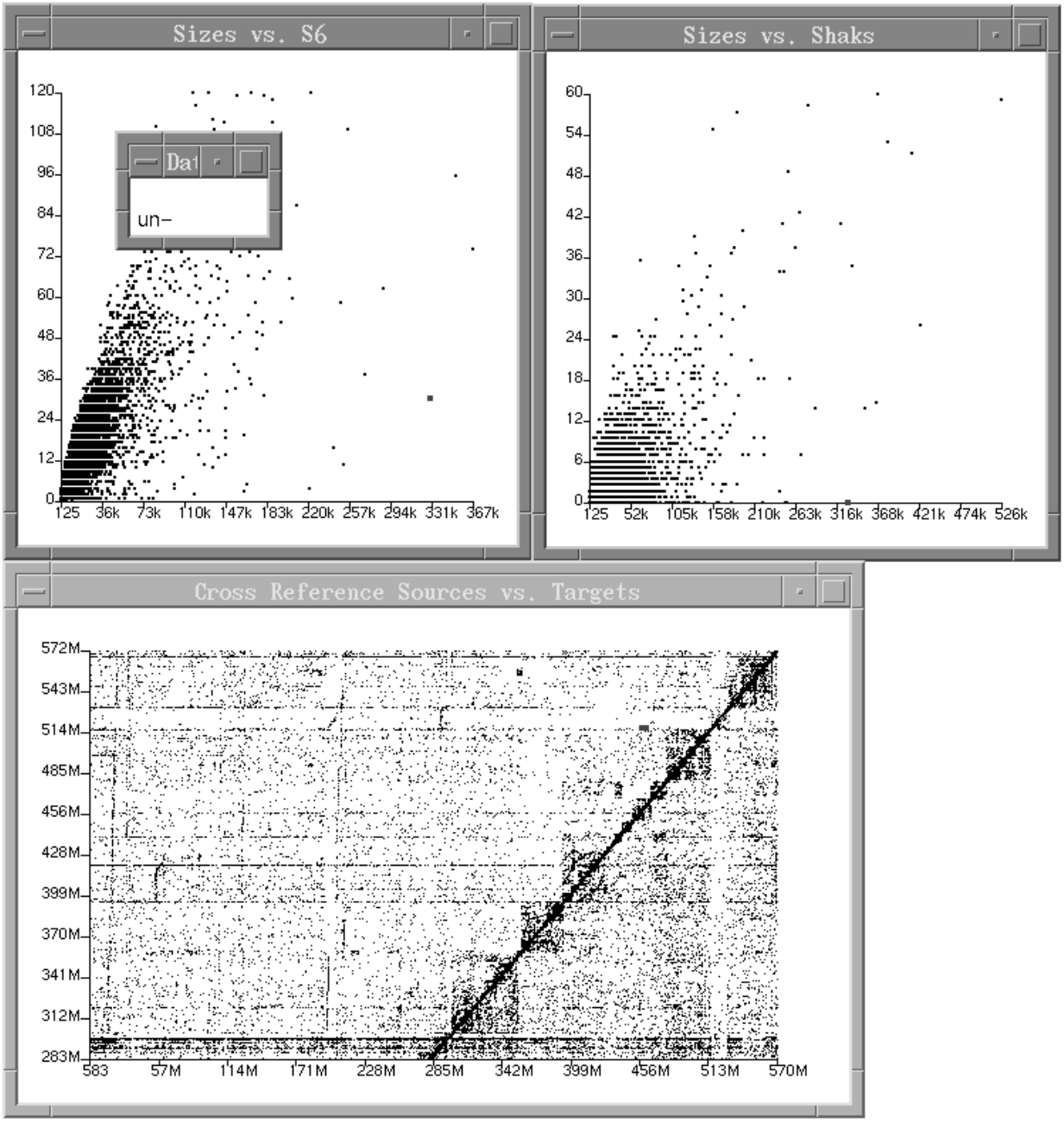


Figure 7: Pipelined visualizations with query results.

intersection, difference, and complement—to be used at any point in the sequence. By this means a user can construct a logically complex query with a few simple pointer motions. A hypothetical query appears in Figure 6; the user has specified the union of three bounding box queries, followed by logical difference with a radial query (note the 'bite' taken out of the upper right hand corner of one of the boxes).

The result of a set identification query is a subset of the original data that the visualizer outputs as a set of numbers. Like the original data, this result itself may be hard to understand unless it is properly displayed. The visualizer addresses this problem by permitting pipelining of individual instances, so that the output of one visualization can be treated as the input of another. Pipelining, along with several other features discussed so far, are presented in Figure 7. Three plots have been pipelined together: entry size vs. number of sense-6 tags, entry size vs. number of quotations from Shakespeare, and cross-reference sources vs. targets. Zooming has been used on both the size/sense-6 plot and the cross reference plot to magnify the region of interest. The point identification window is used in the size/sense-6 plot to indicate that the entry of interest is **un-**; note that this entry is one of the extreme outliers. This selection has been passed on to the other two graphs, where we learn that the entry possesses two other important characteristics. First, it is the largest entry without a quotation from Shakespeare (note the position of the point on the x-axis). Second, it is a cross-reference target for much of the dictionary, since it corresponds to one of the horizontal lines in the cross-reference plot.[2]

Pipelined visualizations have several interesting properties. They are useful for exploring the characteristics of outliers; one can quickly determine if exceptional points in one plot are also exceptional in another. They also serve as a unique kind of multi-dimensional interface to data; instead of combining all dimensions in a single plot, they are shown as a sequence of dimension pairs, with the pipeline serving as the unifying connection for querying. Pipelines also separate the output and input 'vocabularies'; we can select points in a plot where it is most convenient to do so, but observe the properties of those selected points in another plot, where clusters may be more salient. Finally, pipelining need not be confined to the visualizer; the output can also be used by other tools that query or display the text[17].

3 Related software.

The visualizer described in this paper is similar to several other programs for graphical display of data. It is worthwhile to consider them in some detail.

One class of related software is that of graphical statistical packages, such as SYSTAT, SAS, and S[18]. These programs are designed to automate the drudgery of statistical calculation. They are typically used by statisticians, who are interested in computing measures of central tendency and variance, and then in making these com-

[2] On a computer monitor, these points appear as red squares, and so stand out much better than in this monochrome reproduction. The third line from the top of the cross-reference plot (at about 514M) corresponds to **un-**.

putations easily and quickly understood. Statistical packages provide functions such as means, modes, quantiles, standard deviations, time series, regression models, and so on. Some packages, like S, have extensive facilities for plotting bar charts, histograms, pie charts, probability density charts, and arbitrary graphs. They are at their best in generating static, two-dimensional plots of moderate amounts of data.

A second class of related software is that of scientific visualization packages, including AVS, PV-Wave, and IBM's Data Visualizer. These programs are used to create simulations of real-world objects and then to view them from different perspectives. Scientific visualizers are typically used by scientists, engineers, or doctors, for whom studying a simulation is less costly or difficult than studying the phenomena directly. Scientific visualization packages provide such functions as signal processing, image processing, volume visualization, and animation. Some packages also contain an extensive library for numerical and symbolic work, including interpolation, transforms, and differential analysis. They are at their best in generating multidimensional displays of large amounts of data that are time-varying or can be animated.

A third class of related software is that of visual interfaces to databases. This includes systems such as the Spatial Data Management System[19], Media Room[20], and interfaces for simplified boolean querying[21,22], but the most ubiquitous member of the category is the spreadsheet. A spreadsheet is essentially a collection of data dependencies whose individual data elements can be laid out in an informative fashion[23]. Spreadsheets are typically used by business people to answer 'what if' questions; a hypothetical set of input data is entered, and the sheet computes the projected result. Spreadsheets are also simulators, though usually of intangible phenomena. Standard spreadsheets incorporate basic arithmetic and statistical functions, and usually some accounting functions. Advanced spreadsheets support more than two dimensions, include a wide range of presentation graphics, and can be interconnected with each other and with standard enterprise databases.

Each of these types of software has its particular strengths and weaknesses. Spreadsheets do not provide sophisticated mathematics, but do not require it, either. Statistical programs are good for two-dimensional data sets, but usually do not support animation. Scientific visualizers can handle large multidimensional data sets, but often require that the data be modelled outside the system—that is, a scientific visualizer can display shapes, but it is up to the user to create these shapes from the data. Scientific visualizers also make heavy demands on the computing environment. Spreadsheets have slightly more sophisticated graphical input methods than statistical or scientific visualizers, since they allow the graphical selection of rows and columns of the data, but this facility is usually only available for editing the data's schema, and not its instance. Spreadsheets also tend to have less sophisticated layout facilities.

The visualizer described in this paper draws from each of these classes of software, and makes its own modest contributions. It is not simply a statistics package, because the query mechanisms are more powerful and seem more relevant to visualizing texts. It is not simply a visual boolean querying mechanism, since provides for interconnection of plots and the viewing of results across plots. It is not as demanding of the user as

a scientific or statistical visualizer, but it can assist with detailed analysis. It shares with spreadsheets the ability to access databases; and like advanced spreadsheets, the visualizer employs the notion of interconnecting multiple instances of the same tool to handle multi-dimensional problems.

4 Discussion and future work.

It is interesting to contrast the computational properties of textual and (putatively) visual systems. Traditionally, we distinguish between the computational expressiveness of a system (that is, the set of problems it can solve) and its complexity (that is, the efficiency with which it solves a given problem). The computational expressiveness of the visualizer is the same as that of textual systems. Any graphical query we can submit using the text visualizer has a textual equivalent, and vice versa. The hypothetical graphical query of Figure 6, for example, could also have been posed as follows:

$$
\begin{aligned}
&result = entry \leftrightarrow \\
&((xr.source > 307M \wedge xr.source < 479M \wedge xr.sink < 134M \wedge xr.sink > 124M) \vee \\
&(xr.source > 436M \wedge xr.source > 444M \wedge xr.sink > 89M \wedge xr.sink < 235M) \vee \\
&(xr.source > 439M \wedge xr.source < 526M \wedge xr.sink > 214M \wedge xr.sink < 299M)) \wedge \\
&\neg \left(\sqrt{\left| \frac{xr.source - 526M}{572M} \right|^2 + \left| \frac{xr.sink - 299M}{572M} \right|^2} < C \right)
\end{aligned}
$$

While the expressiveness of the two systems is the same, there is a clear difference in the complexity of specification. The query in Figure 6 is quite simple to specify graphically, although somewhat imprecise. The textual equivalent is very precise, but is so complicated that few users would bother to pose it (and fewer still would do so correctly). Moreover, the textual query is highly improbable in any case, since without the visualization, the user is not aware of the groupings that would give rise to such a query.

The precision of the graphical query could be improved by providing the user with direct feedback about the bounds of the bounding box, so that it could be drawn more accurately. This would be similar to the way that window systems provide feedback on the size and shape of a window that is being stretched or repositioned. Another way to improve precision is to use 'gravitation' towards a pre-specified grid. A third technique is to simply permit direct input of the bounds of the box. This technique is acceptable for highly regular figures, but less suitable for arbitrary polygons.

Graphical queries are not always simpler. 'Find every cross-reference whose source is an odd number' is easily specified in a textual query language, but if done graphically, would require approximately 286 million bounding boxes. Thus, each language makes some queries succinct and others complex. The tradeoff results from the different types of groupings the two systems provide.

It would be useful to have a mathematical expression of our informal notion of query complexity. Such a model might define a complexity measure based on the minimal

length of a query, given the conditions required of the solution. Thus, for the query corresponding to Figure 6, the textual language is more complex because it is longer. A more elaborate version of the same model would identify classes of queries whose lengths were related to the size of the initial conditions by a specific class of function—for example, logarithmic, polynomial, or exponential. The model might also be augmented with behavioural information about the difficulty that people have with various types of logical constructs. People often use negation wrongly, for example, so it might be reasonable to count extra 'length' in queries that use negation[24].

The equivalence in computational expressiveness of the two systems raises the old problem: by virtue of what characteristic do we refer to one language as visual, and the other as textual? It is difficult to evaluate any data visualizer without some appreciation of what it means to be visual, but text visualizers are particularly vulnerable, since by definition they (purport to) straddle the two types of inscription.

I am partial to Goodman's notion of density as the fundamental determiner of visual languages[25]. By 'density', Goodman means that there is no finite level of distinction that is sufficient to distinguish all pairs of values. Bertin also recognizes the importance of this distinction, if not its primacy[26], and other writers have fastened on the complementary notion of discreteness as the essential element of writing[9]. Unlike many of the other suggested lines of demarcation, the density-discreteness criterion has implications for the computational complexity of languages. It can be shown that analog computers can compute certain problems much more quickly than can digital computers[27]. This lends support to the notion that our intuition about the effectiveness of visual query languages is due to the complexity advantages to be found in dense symbol systems.

The visualizer described here is under continuing development. It will serve as a testbed for work on new query mechanisms, and as a prototype for studying the use of these queries in making sense of words. The next step is to conduct some practical tests to discover the virtues and faults of the software, and to suggest new ways that we can interact with visualizations to make sense of words.

5 Acknowledgments.

Financial support was provided by an IBM Canada Research Fellowship and by the Natural Sciences and Engineering Research Council of Canada. I am grateful to Nick Cooper of IBM for supporting this line of inquiry, and to Frank Tompa, who is a constant source of ideas and encouragement.

6 References

1. E.H. Gombrich, *Art and Illusion: A Study in the Psychology of Pictorial Representation,* Princeton University Press, Princeton, New Jersey (1969).

2. N. Goodman, *Languages of Art,* Hackett Publishing Co. (1976).

3. S. K. Langer, *Philosophy in a New Key: A Study in the Symbolism of Reason, Rite, and Art (3rd edition),* Harvard University Press, Cambridge, Massachusetts (1957).

4. E. Burke, *A Philosophical Enquiry into the Origins of our Ideas of the Sublime and the Beautiful,* Routledge & Kegan Paul, London, England (1958).

5. G. Lessing, *Laocoon: an Essay on the Limits of Painting and Poetry,* Bobbs-Merrill, Indianapolis, Indiana (1962).

6. J. H. Larkin and H. A. Simon, "Why a Diagram is (Sometimes) Worth Ten Thousand Words," *Cognitive Science* **11** pp.65-100 (1987).

7. R. Arnheim, *Art and Visual Perception: A Psychology of the Creative Eye,* University of California Press, Berkeley, California (1974).

8. M. McLuhan, *Understanding Media: The Extensions of Man,* New American Library, New York, New York (1964).

9. R. Harris, *The Origin of Writing,* Gerald Duckword & Co., London, England (1986).

10. J.E. Littlewood, *A Mathematician's Miscellany,* Methuen & Co. Ltd., London, England (1957).

11. R. H. Mendez, *Visualization in Supercomputing,* Springer-Verlag, New York, N.Y. (1990).

12. *Visualization in Teaching and Learning Mathematics,* Mathematical Association of America (1991).

13. D.L. Berg, *A Guide to The Oxford English Dictionary,* Oxford University Press, Oxford, England (1993).

14. D. R. Raymond and F. Wm. Tompa, "Hypertext and the Oxford English Dictionary," *Communications of the ACM* **31**(7) pp.871-879 (July 1988).

15. W. C. Brinton, *Graphical Methods for Presenting Facts,* Engineering Magazine Company, New York, N.Y. (1914).

16. R. L. Burden and J. D. Faires, *Numerical Analysis,* PWS-Kent Publishing Co., Boston, Massachusetts (1989).

17. D. R. Raymond, "Flexible Text Display with Lector," *IEEE Computer* **25**(8) pp.49-60 (August 1992).

18. R. A. Becker and J. M. Chambers, *S: An Interactive Environment for Data Analysis and Graphics,* Wadsworth Inc., Belmont, California (1984).

19. C. F. Herot, "Spatial Management of Data," *ACM Transactions on Database Systems* **5**(4) pp.493-514 (December 1980).

20. N. Negroponte, "Media Room," *Proceedings of the Society for Information Display* **22**(2) pp.109-113 (1981).

21. A. Michard, "Graphical Presentation of Boolean Expression in a Database Query Language: Design Notes and an Ergonomic Evaluation," *Behaviour and Information Technology* **1**(3) pp.279-288 Taylor & Francis Ltd., (1982).

22. M. D. Williams, "What Makes RABBIT Run?," *International Journal of Man-Machine Studies* **21** pp.333-352 (1984).

23. A. Kay, "Computer Software," *Scientific American* **251**(3) pp.53-59 (September 1984).

24. W. S. Cooper, "Exploiting the Maximum Entropy Principle to Increase Retrieval Effectiveness," *Journal of the American Society for Information Science* **34**(1) pp.31-39 John Wiley & Sons, Inc., (January 1983).

25. D. R. Raymond, "Characterizing Visual Languages," *IEEE Workshop on Visual Languages*, pp.176-182 (October 9-11, 1991).

26. J. Bertin, *Semiology of Graphics,* University of Wisconsin Press, Madison, Wisconsin (1983).

27. A.K. Dewdney, "Analog Gadgets that Solve a Diversity of Problems and Raise an Array of Questions," *Scientific American*, pp.18-29 (June 1985).