Fax: An Alternative to SGML

Kenneth W. Church, William A. Gale, Jonathan I. Helfman and David D. Lewis

AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, NJ 07974, USA
kwc@research.att.com

We have argued elsewhere (Church and Mercer, 1993) that text is more available than ever before, and that the availability of massive quantities of data has been responsible for much of the recent interest in text analysis. Ideally, we would hope that this data would be distributed in a convenient format such as SGML (Goldfarb, 1990), but in practice, we usually have to work with the data in whatever format it happens to be in, since we usually aren't in much of a position to tell the data providers how to do their business. Recently, we have been working with a collection of 15,000 AT&T internal documents (500,000 pages or 100 million words). Unfortunately, this data is stored in a particularly inconvenient format: fax.

It might seem odd to work with a corpus of faxes, but faxes might well be the way of the future. Fax is used a lot more than SGML (especially over telephone networks). SGML might be more convenient for our research, but the world is using fax.

So, what can we do with a corpus of faxes? Right now, we might not consider a fax to be as ''machine readable'' as a text file, but if we set our minds to it, it ought to be possible to do practically anything with a fax that can be done with a text file. In particular, it should be possible to search (grep) for sections of interest in a fax, cut them out of one document and paste them into another, or use them as input to an arbitrary program. If we are successful, the user shouldn't have to know about markup languages (e.g., SGML), tables, figures, floating displays, headers, footers, footnotes, columns, fonts, point sizes, character sets (e.g., ascii, unicode), and all sorts of other ''technical details.'' As far as the user is concerned, the system is just faxes (or bitmaps), through and through.

1. Image EMACS: the Ultimate in WYSIWYG

Many of the pieces of this proposal are well underway. The Image EMACS editor (Bagley and Kopec, 1992; Bush, 1993), for example, makes it possible to edit bitmaps more or less the same way that one edits a text file. You can scan an image into the computer, change a few words, re-justify a paragraph, and then print it out again.

Image EMACS is the ultimate in WYSIWYG: what you see is what you get, *and vice versa*. Most WYSIWYG editors do only half the job; they let you print out what you see, but they don't let you scan it back in. The round trip is key. It makes it possible to work with any document in any format. (At worst, the document can be printed out and scanned into Image EMACS.) Most WYSIWYG editors don't provide a complete round trip and therefore their applicability is limited to a relatively small fraction of the world's documents, those that happen to be formatted in a suitable markup language.

2. Fax-a-Query: the Ultimate in WYSIWYG Interfaces for Information Retrieval (IR)

Users will need to search bitmaps for sections of interest. Traditionally, most IR systems have been developed for collections of text files rather than bitmaps. The user types in a query and the system retrieves a set of matching documents. Some of these systems depend on manual indexing, e.g., subject terms or hypertext links. Others allow the user to type in an arbitrary piece of text as input. Documents are retrieved by matching words against the query and weighting appropriately (Salton, 1989).

These systems have been extended to retrieve bitmaps, by first pre-processing the bitmaps with an OCR program. Although the OCR results are far from perfect, and users would complain about the OCR errors if they saw them, the OCR output has been shown to be more than adequate for retrieval purposes (Smith (1990), Taghva *et al* (to appear)).

But why should a user have to type in a query? Why not provide a complete round trip capability? If OCR were used on the queries as well as on the documents, then the query could be a page of a book, article, a fax, or whatever. As far as the user is concerned, the system is just faxes (or bitmaps), through and through.

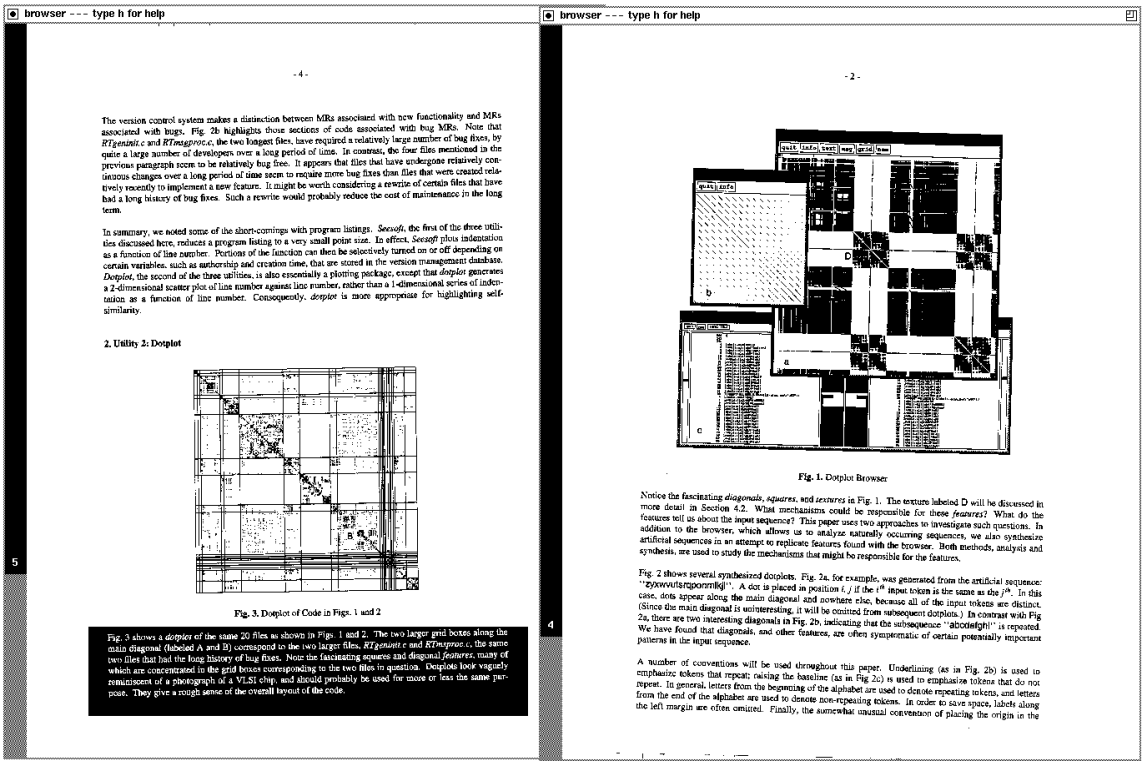

Figure 1: An example of the fax-a-query prototype. A user is reading a document in a bitmap browser (left panel), and comes across a topic of interest. The user sweeps a box over an interesting section of the bitmap (inverse video at bottom of left panel), which causes the corresponding words (produced by OCR) to be sent to an information retrieval system. A relevant document pops up in another bitmap browser (right panel).

We call this proposal Fax-a-Query, and illustrate it in Figure 1. A user is reading a document in a bitmap browser, and comes across a topic of interest. The user sweeps a box over an interesting section of the bitmap, which causes the corresponding words (produced by OCR) to be sent to an information retrieval system. A relevant document pops up in another bitmap browser.

Fax-a-Query is also useful for retrieving pictures as well as text. Most picture retrieval system require manual indexing, which can be very expensive. However, since a picture is often surrounded by useful text such as a caption, one can find the picture by matching on the text.

We have applied a prototype Fax-a-Query system to our database of 15,000 AT&T internal documents. These documents were scanned into the computer by the AT&T library for archival purposes. They are stored in TIFF format at 400 dots per inch, using Group 4 fax compression. It took us about a minute per page or a year of real time to OCR the collection and 40 hours of real time to index the collection with the SMART information retrieval system (Salton and McGill, 1983, chapter 4).[1] The bitmap browser was borrowed from the Ferret system (Katseff, personal communication).

Fax-a-Query was also designed to be usable from a standard fax machine, for users that may be on the road and don't have access to a terminal with a window system. A user could fax a query to the system and the system would fax back some

1. The OCR errors slow the indexing process considerably since they make the vocabulary too large to fit in main memory. Our data has a huge vocabulary (3 million words), most of which are OCR errors. By comparison, the TREC text collection (Dumais, 1994) has a much smaller vocabulary (1 million words). The difference in vocabulary sizes is especially significant given that TREC is considerably larger (2 gigabytes) than our OCR output (1 gigabyte).

relevant documents. In this way, a user could call the home office from any public fax machine anywhere and access documents in a fax mailbox, a private file computer, or a public library. (This capability is currently limited by the fact that OCR doesn't work very well on low resolution faxes.)

3. Do We Need OCR?

Fax-a-Query makes heavy use of OCR, but does so in such a way that users are often unaware of what is actually happening behind the scenes. Image EMACS works directly on the pixels, in order to avoid OCR errors. Even though users can be fairly well shielded from the limitations of the OCR program, the OCR errors are frustrating nonetheless.

Two examples of the word ''pair'' are shown in Figure 2. Both examples were extracted from the same document, but from different pages. One of them was recognized correctly and the other was misrecognized as ''liair''. As can be seen in Figure 2, the two images are almost identical. Even a very simple-minded measure such as Hamming distance would have worked better than OCR, at least in this case.

The ''liair'' error was probably caused by incorrectly segmenting the ''p'' into two letters, and then labeling the left half of the ''p'' as an ''l'' and the second half as an ''i''. This error is particularly inexcusable since the spacing of the letters within a word is completely determined by the font. There is no way that ''li'' should be confusable with ''p'' since it would require shifting the ''l'' with respect to the ''i'' in both the horizontal and vertical dimensions in ways that are extremely unlikely. The Hamming distance approach would not make this kind of error because it works at the word-level rather than the character-level, and so it would not try to shift parts of words (or letters) around in crazy ways.

In general, we have found that two instances of the same word in the same document are often very similar to one another, much more so than two instances from different documents. Figure 3, for example, shows a number of examples of the word ''using'' selected from two different documents. If we sum all of the instances of ''using'' across the two documents, as shown in the bottom-most panel, we get a mess, indicating that we can't use Hamming distance, or anything like it, for comparing across two documents. But if we sum within a single document, as shown in the two panels just above the bottom-most panel, then we find much better agreement, indicating that something like Hamming distance ought to work fairly well, as long as we restrict the search to a single document.

The strong document effect should not be surprising. Chances are that all of the instances of ''using'' have been distorted in more or less the same way. They were probably all Xeroxed about equally often. The gain control on the scanner was probably fairly consistent throughout. The font is likely to be the same. The point size is likely to be the same, and so on. Some authors refer to these factors as defects (Baird, 1992), but we prefer to think of them as document-specific properties.

We have used this Hamming distance approach to build a predicate that compares two boxes and tests whether the pixels in the two boxes correspond to the same word. In the case of the two ''pairs'' in Figure 2, for example, the predicate produces the desired result. This distance measure has been used to implement a search command. When the user clicks on an instance of a word, the system highlights the next instance of the same word, by searching the bitmap for the next place that has almost the same pixels.[2]

It is remarkable that this search command manages to accomplish much of what we had been doing with OCR, but without the C (it is word-based rather than character-based) and without the R (it doesn't need to recognize the words in order to search for the next instance of the same thing). This opens an interesting question: how much natural language processing can be done without the C and without the R? For example, could we count ngram statistics at the pixel-level without giving the OCR program a chance to screw up the Cs and the Rs?

4. Conclusions: Bitmaps are The Way of The Future

We have been working with a large corpus of faxes (15,000 documents or 500,000 pages or

2. It is possible to implement this search much more efficiently by pre-computing a few moments for each of the words in the bitmap and using these moments to quickly exclude words that are too big or too small, or too spread out or not spread out enough.

100,000,000 words). Faxes raise a number of interesting technical challenges: we need editors, search engines, and much more. Of course, we wouldn't have to work on these hard problems if only people would use SGML. But, people aren't using SGML. SGML may be more convenient for us, but the world is using fax because it is more convenient for them.

Fax hardware and software are everywhere: hotels, airports, news stands, etc. Everyone knows how to use a fax machine. Word processors are more expensive, and require more training and skill. The markup issues, for example, are very demanding on the users. Part of the problem may be the fault of the markup languages, but the real problem is that the concepts are just plain hard. Most users don't want to know about tables, figures, floating displays, headers, footers, footnotes, columns, fonts, point sizes, character sets, and so on.

Libraries are scanning large numbers of documents because scanning has become cheaper and more convenient than microfiche. Our library is scanning 10^5 pages per year. Our library has also been trying to archive ''machine readable'' text files in addition to the bitmaps, but with somewhat less success. Because it is too expense to re-key the text, they have been asking authors for text files, but most authors aren't very cooperative.

Even when the text file is available, we should also archive the bitmap as well, because the bitmap is more likely to survive the test of time. We tend to think of the text file as the master copy, and the bitmap and the hardcopy as a by-product, when in fact, it should probably be the other way around. When the first author was finishing his Ph.D., he had to generate a copy of the thesis for archival purposes. At the time, it seemed that the school library was stuck in the stone age, because they insisted on a hardcopy printed on good paper, and they were not interested in his wonderful ''machine readable'' electronic version. In retrospect, they made the right decision. Even if the tapes had not rotted in his basement, he still couldn't read them because the tape reader is long gone, and the tape format is now obsolete. The markup language is also probably dead (does anyone remember *R*?), along with the computer (a PDP-10), the operating system (ITS), and most other aspects of the hardware and software that would be needed to read the electronic version.

The debate between text files or bitmaps is analogous to the old debate between character-based terminals such as a VT100 and bitmap terminals. At the time, bitmap terminals seemed wasteful to some because they required what was then a lot of memory, but nowadays, it is hard to find a character-based terminal anywhere, and it is hard to remember why anyone would have wanted one. How could you run a window system on such a terminal? How could you do any interesting graphics? There were solutions, of course, but they weren't pretty.

So too, there might soon be a day when people might find it hard to imagine why anyone would want a text file. How could you do any interesting graphics? Equations? There are solutions (markup and include files), but they aren't pretty. Of course, bitmaps require a little more space (a 400 dpi G4 fax takes about 20 times the space as the equivalent text file), but the bitmap is so much more powerful and so much easier to use that it is well worth the extra space.

References

Bagley, S. and Kopec, G. (1992) ''Editing Images of Text,'' Xerox, PARC.

Baird, H. (1992) *Document Image Defect Models*, in Baird, Bunke and Yamamoto (eds.) *Structured Document Image Analysis*, Springer-Verlag, Berlin, Germany, pp. 546-556.

Bush, M. (1993) *Speech and Text-Image Processing in Documents*, ARPA Human Language Technology, Morgan Kaufmann Publishers, San Francisco, CA, USA., pp. 376-380.

Church, K. and Mercer, R. (1993) ''Introduction to the Special Issue on Computational Linguistics Using Large Corpora,'' *Computational Linguistics*, 19:1, pp. 1-24.

Dumais, S. (1994) ''Latent Semantic Indexing (LSI) and TREC-2,'' in Harman, D. (ed.) *The Second Text REtrieval Conference (TREC-2)*, National Institute of Standards and Technology, Gaithersburg, MD, USA.

Goldfarb, C. (1990) ''The SGML Handbook,'' Clarendon Press.

Salton, G. and McGill, M. (1983) *Introduction to Modern Information Retrieval*, McGraw-Hill Book Company, New York, NY, USA.

Salton, G. (1989) *Automatic Text Processing*, Addison-Wesley Publishing Co., Reading, MA, USA.

Smith, S. (1990) ''An Analysis of the Effects of Data Corruption on Text Retrieval Performance,'' Thinking Machines Corp., DR90-1, Cambridge, MA, USA.

Taghva, K, Borsack, J. and Condit, A. (to appear) ''Results of Applying Probabilistic IR to OCR Text,'' in Seventeenth International ACM SIGIR Conference on Research and Development in Information Retrieval.

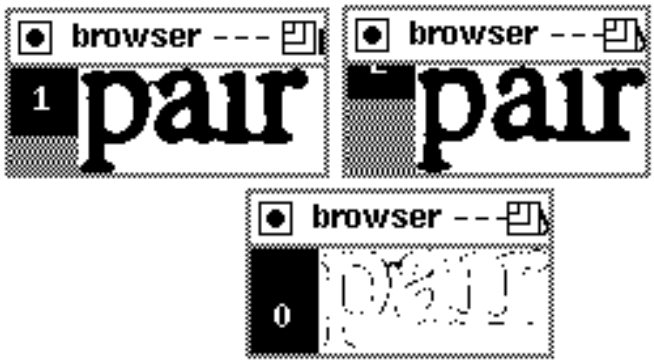

Figure 2: Two instances of the word ‘‘pair’’ and their pixel-wise differences. The pixel-wise differences show that the two images are almost identical, and yet, one was recognized correctly as ‘‘pair’’ and the other incorrectly as ‘‘liair’’. Even a very simple-minded measure such as Hamming distance would have worked better than OCR, at least in this case.

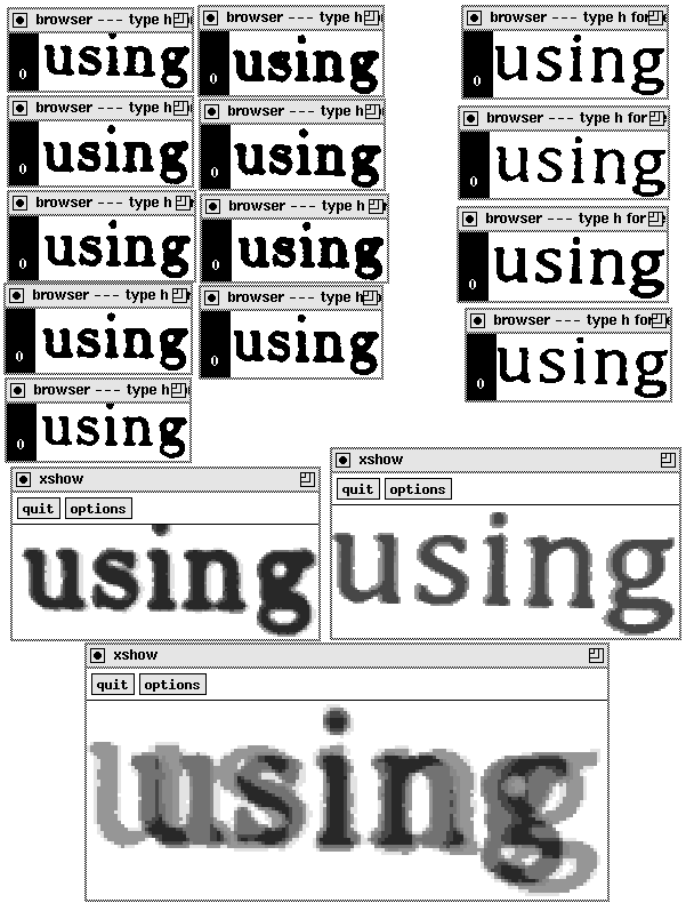

Figure 3: Hamming distance is much more appropriate within documents than across documents. The upper left shows 9 instances of ‘‘using’’ extracted from one document and the upper right shows 4 more instances extracted from another document. The 9 instances are summed into one image (middle left) and the 4 instances are summed into another image (middle right). These two images (middle left and middle right) indicate a high degree of internal consistency within a document. The bottom image is the sum of all 13 instances. Clearly, there is more consistency within documents than across documents.