

Treetables and Other Visualizations for Email Threads

Paula S. Newman

ABSTRACT

In this paper, we describe some new visualization methods for email threads. The methods concatenate initial message texts, or full texts shorn of extraneous material, into logical groupings embedded in, or closely aligned with, thread structure representations. The results are intended to provide useful thread overviews, and to enable coherent, efficient reading of thread content.

Keywords

Email archives, email threads, computer mediated communication, tree visualization, trees, narrow trees, treetables, email message analysis, threading, overview + detail, focus + context

INTRODUCTION

An email archive records the discussions of a virtual community made up of the subscribers to a discussion list. The subscribers may represent an administrative unit, a working group (e.g., a standards group), or just a group of people with a common interest (e.g., motorcycle owners, or users of a programming language). Archives are read by:

- New working group participants, to understand group operations and concerns.
- Active list participants, to review prior or current discussions.
- Inactive list participants, to maintain awareness of list activity.
- Non-participants, to find information, to understand the background of a work-product, or for purposes of sociological, legal, or historical research.

As the popularity of email lists has grown, so has the need for more powerful tools to assist in the exploration and reading of their archived content. One area in which conventional tools are inadequate is in the presentation of longer threads. Figure 1 illustrates the distribution of thread lengths greater than 2 messages for an excerpt of the rec.motorcycle newsgroup covering about a month. While most threads are relatively short, there are many long threads often consisting of several simultaneous sub conversations.

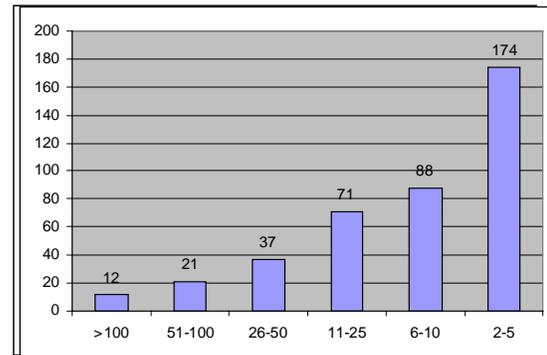


Figure 1. Thread lengths in rec.motorcycle excerpt

These are potentially important threads, as suggested by the number of contributions, but their structural complexity makes them difficult to represent in ways that allow efficient assimilation of their content.

The MailContent project is concerned with developing an integrated system for email archive exploration and reading. The current system includes facilities at both the corpus and thread review levels, enabling a gradual descent from general corpus exploration to in-depth reading of threads, in whole or in part.

In this paper we focus on the facilities at the thread level, in which two new, alternative, tree visualizations are used for both thread overviews and reading assistance. Both visualizations are founded on the principle of providing content within the context of thread structure, so that conversations can be followed in a natural way. We begin by discussing current approaches to thread visualization and then move to the MailContent approaches.

READING THREADS TODAY

For a very long period, a standard means of presenting archived threads has been via variations of Hypermail [1], ultimately deriving from Intermail [2]. This approach represents the tree structure of a thread by indentation, with each message represented by an author name and date, linked to the full message, and followed (recursively) by an indented list of the responses to that message. Message-embedded links representing the tree structure are also provided, so that the thread sequence can be followed in either way. But either usage requires that messages be selected and read one-by-one. For all but the shortest threads, this means that following the complex structure of a thread is a slow, fragmented, difficult process.

Within the last year or so, at least one commercial list-hosting web site has addressed this issue by

concatenating the messages of a thread into single documents (in groups of 10 messages), to allow more linear, continuous reading. In the initial approach used by Google Groups [3], the messages were concatenated in date order. This ordering is inadequate for complex threads, as it replaces the physical fragmentation of the Hypermail approach with the conceptual fragmentation arising from interspersed sub-conversations. Also, because the messages are conjoined, the reader must continually skip and scroll over the large amount of extraneous information in messages, such as quotes of entire prior messages, and various kinds of contact information. However, the extraneous information is highlighted, presumably to assist in skipping.

A further step was taken by Google within a few weeks of this writing (September 2001), adding an “overview + detail” style (Plaisant et.al. [4]) framed presentation. The left-hand frame contains an indented tree listing message numbers and names, usable for navigation and to maintain context. Successive indentations are one character in width, to allow the outline to fit into the frame, and are augmented by prefixed symbols to clarify the nesting depth. The right hand frame contains the concatenated messages in the same order as in the left hand frame, but not indented.

This step begins to approximate the kind of thread visualizations needed, namely, the presentation of message texts combined in message-response order, while at the same time maintaining user awareness of the place of the message sequence being read within the overall thread context. The goal is, essentially, to make reading threaded conversations as convenient as reading a theatrical script.

A few more experimental approaches to thread presentation that relate to the methods to be described below should also be noted. First, a radical attempt at message concatenation is described by Yabe et.al. [5], in which the text of a message is interspersed with responses to particular passages. While the approach founders on logistic problems, such as the fragmentation of logically-related text by inserted responses, it does illustrate an attempt to more adequately account for the conversational aspects of threads. Two other approaches use versions of classic node-plus-edge tree representations to represent threads. Smith and Fiore [6] illustrate threads by classic trees with very tiny nodes. The nodes are insufficient to contain any identifying text, but the representation conveys the structure of the conversation rather clearly. Also, in a message input approach described by Popolov et.al [7], the user views a lattice representing the state of the thread, and then identifies the nodes to which the input is a response.

Thread Overviews

We move now to another, essentially unrecognized, problem presented by longer threads, namely the lack of overviews conveying both structure and content. The longer the thread, the more likely it is to branch into a number of subtopics. So a facility helping the reader to decide whether to read a thread, and, if so, what parts, would be useful.

An ideal approach to an overview, not yet achievable, would consist of a summary of thread structure and content, identifying the subtopics covered, the types of opinions expressed, etc. This is quite difficult to accomplish; current summarization methods do not generally deal with conversations.

Instead, the closest approximations to thread overviews we have today are indented author listings, and the ability to view initial message fragments (but not within an indented listing) as in Outlook. Unfortunately, the utility of the fragments as indicators of message content is marred by the fact they are simply the first text elements following the message header, which are often quoted excerpts, for example: “On October 5 2000 John Smith wrote: >I’ve been on vacation since last week and therefore did not see your messages”.

MAILCONTENT THREAD VISUALIZATIONS

The presentation of threads in MailContent evolved in two major stages, resulting in two new methods of visualizing trees called, respectively, “narrow trees” and “treetables”, used for both thread overviews and for deeper exploration.

While the methods were developed in connection with email threads, they are also suited to the exploration of other trees with similar properties, namely: (a) that not just the leaves, but also the interior nodes of the trees represent significant amounts of information, and (b) that the paths within the tree are very meaningful.

The two visualizations are described below in the order in which they were developed, and their respective affordances compared. Both methods rely heavily on message analyses that isolate the essential text content of a message from extraneous material, and on detailed threading. These underlying processes are discussed briefly following the visualization discussions.

Narrow trees

Our first approach to thread visualization proceeded directly from the challenge of allowing semi-linear reading of multiple messages in the context of a vertically represented thread structure. The “narrow tree” representation developed for this purpose strongly limits indentation by the basic device of indenting a message successor only if there is more than one

successor. The result allows embedded texts to be read "down the page" in the context of the structure.

An example "narrow tree" embedding the initial texts of each message, for use as a thread overview, is shown in figure 2. Messages 5045 and 5114 are the direct responses to the initial message 5037. Message 5196 is the only response to 5114, and message 5254 the only response to 5196, etc.

This thread overview has a maximum indent level of 1. Furthermore, since messages 5114 through 5873 constitute a single chain, their fragments can be scanned linearly. An equivalent, fully indented representation would have a maximum indent level of 7, and the indentations would cause the embedded text to fall far off the end of the window, requiring horizontal scrolling even for this short thread. On the other hand, if the texts were embedded in a structure using tiny indents, as in [3], the structure implied by the indents and 'undents' would be difficult to grasp.

In slightly more detail, a narrow tree structure is built by the following procedure, initially invoked with the first message as the "current message", and 0 as the "current indent level":

1. List the current message at the current indent level.
2. If the current message has only one response, recursively call the procedure for that response, at the current indent level.
3. If the current message has multiple responses, for each response:
 1. add a divider line at the current indent level + 1.
 2. recursively call the procedure for that response at the current indent level + 1;

Informal experimentation with narrow trees indicated that they are useful for several purposes, but also have limitations. They seem to be useful for thread overviews, as illustrated in figure 2, embedding the initial fragments of each message and eliding quotes. Because email messages tend to be brief and to the point, this usage generally provides, via scrolling, a reasonable idea of the themes covered by the thread.

The narrow trees also seemed useful for actual reading of the very short threads that represent a substantial plurality of even high volume lists (see figure 1). Figure 3 illustrates a "reading view" of a short thread that embeds what we will call the "essential text" of the message within the narrow thread structure. The essential text consists of the main text up to an identified closing, interspersed with abbreviations of initial or incorporated first-level quotes.

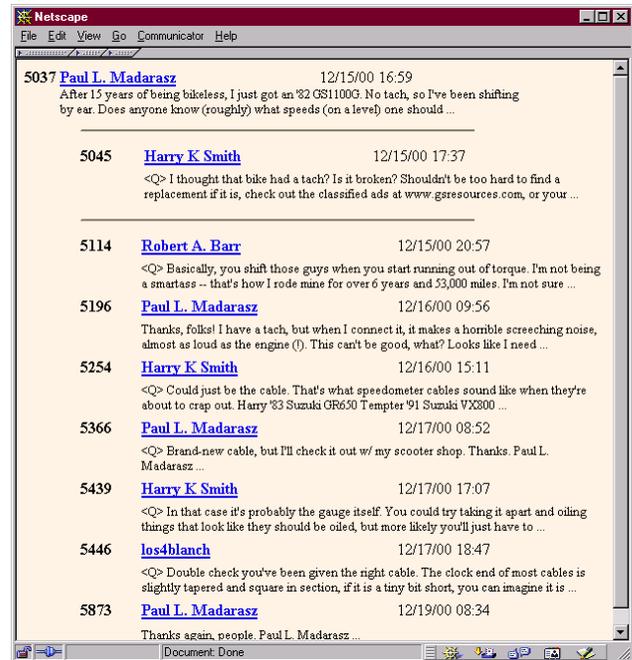


Figure 2. Narrow tree used for thread overview



Figure 3. Narrow tree embedding essential text

For example, figure 4 illustrates the full text of message 451, from which the essential text has been extracted for use as the second message of figure 3.

A closing that terminates the essential text may be a conventional closing (thanks, regards, etc), a signature, a contact block, or a dividing line followed by such information. Initial quotes are included (in abbreviated form) if they are selective (i.e., they do not represent the entire predecessor message), or if the quoting message immediately follows a horizontal bar.

```

On Thu, 30 Nov 2000 06:07:36 GMT, Ray Luce
<murdercycles@earthlink.net> wrote:

> For some strange reason I still don't quite understand, I bought a 99
> aprilia RS50 Replica today. Why would I buy a glorified moped when I
> have a barn full of bikes?
> Maybe you folks can help me figure it out.
> My rational side says, "It stops you from dropping $15,000 on a RSV
> Mille."
> My idiot side says, "It looks so cool and nobody else around has
> one!"
> Does anyone relate?

A carb, arrows exhaust, and some overboring should make it
considerably less boring.
--
-----
Adam Wade          "The only way I'll stop riding
CWRA #4 SDWL #2      is if I stop breathing."
CIMC #1 DoD #2009 LOMP #2      1990 Zephyr 550 (Daphne)
The opinions expressed here are my own, and do not represent
those of my employer in any form.
-----

```

Figure 4. Message 451 of Figure 3

Unfortunately, the same informal experiments that demonstrated some of the benefits of narrow trees also showed that when the reading view occupied more than two screens, which occurs roughly at about 6 messages (or fewer if the contained messages are long), the simple scrolled reading views were inadequate; the thread context was simply lost.

Our first approach to addressing this problem consisted of an “overview + detail” form of narrow thread display, illustrated in figure 5. The left-hand frame provides an overview in the form of a narrow tree outline in one frame (optionally embedding initial message fragments), and a narrow tree reading view in the other. Selecting a message in the outline causes the reading view to be scrolled to that message. Also, if a user is reading a given message, and the predecessor message to which it is a response is not in view, the predecessor can be made to temporarily replace the left-hand frame.

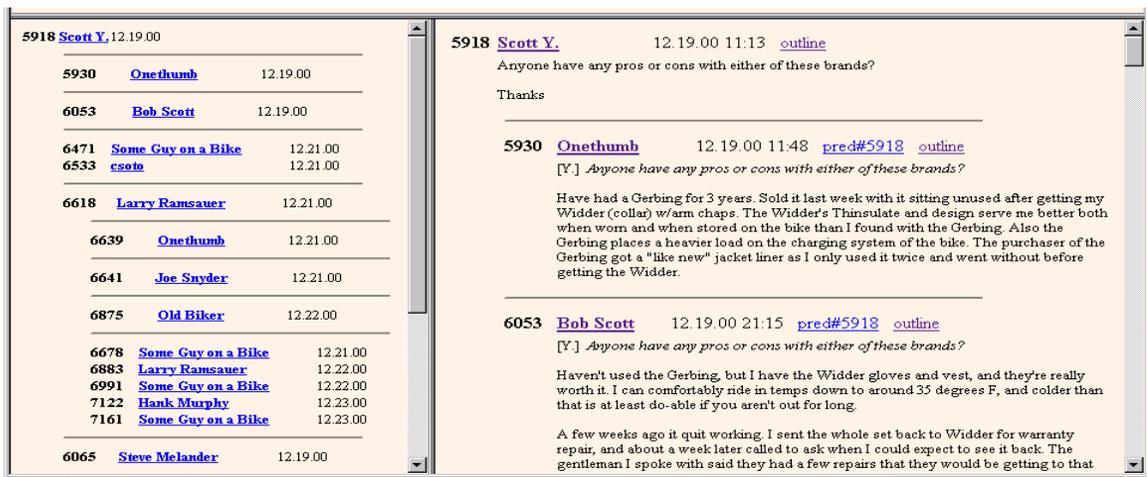


Figure 5. Overview + detail form of narrow tree

Figure 6 shows a treetable-based overview for a medium length thread of 23 messages. The essence of the

Informal tests of this “overview + detail” approach on many threads indicated that its convenience depended on the extent to which: (a) the “overview” could be kept in view without scrolling, and (b) the threads were not too “bushy”, i.e., they did not contain a large number of parallel subthreads. Very bushy threads require both horizontal scrolling, and also too many references back to the overview, to reestablish context, when an “indent” is reached.

TreeTables

While the narrow tree approach thus seemed useful for some thread-overview purposes, and for the reading of shorter threads, its limitations led to search for a more powerful way of presenting threads. Restated, the problems seem to be: (a) the overviews, while they provide a scannable appreciation of general content, do not provide an equal appreciation of structure, especially when any scrolling is involved, and (b) for threads with relatively complex structure, the “reading views” do not meet the challenge of allowing the conversations of a thread to be read and assimilated like those of theatrical scripts.

The search brought us back to considering the classic node-plus-edge tree representation. After all, that representation does show structure, and the play-like reading forms we were seeking could be obtained by views following the root-to-leaf paths in the tree. But the classic representation wastes a lot of space, leaving inadequate room for text. Also, space-efficient methods for displaying large trees tend to intersperse nodes on different paths, making the individual paths hard to isolate. “Treetables” were developed in an effort to avoid those difficulties.

representation is that each column of the table represents a single path from the root of the thread tree to a leaf.

Also, each cell in a row i exactly spans the cells representing its responses in the next row $i+1$. Cell sizes are adjusted to fit into a standard window size, so that while less text is shown for larger trees, most or all of the thread structure can be viewed with very limited scrolling, except in extreme cases. Furthermore, the basic geometry of the treetable ensures that more space is available for text in the bushiest subtrees, which are often those generating the most interest. (In this case we have arbitrarily located those subtrees on the left.)

Like narrow trees, treetables can be used as the base for more detailed explorations of the messages of the thread. One way of performing more detailed explorations, illustrated in figure 7, is adapted from "focus + context" ("fisheye")[4] approaches, in particular ones like TableLens [8], in which the regular structure is preserved, thus limiting difficulties in re-orientation when shifting among focus areas.

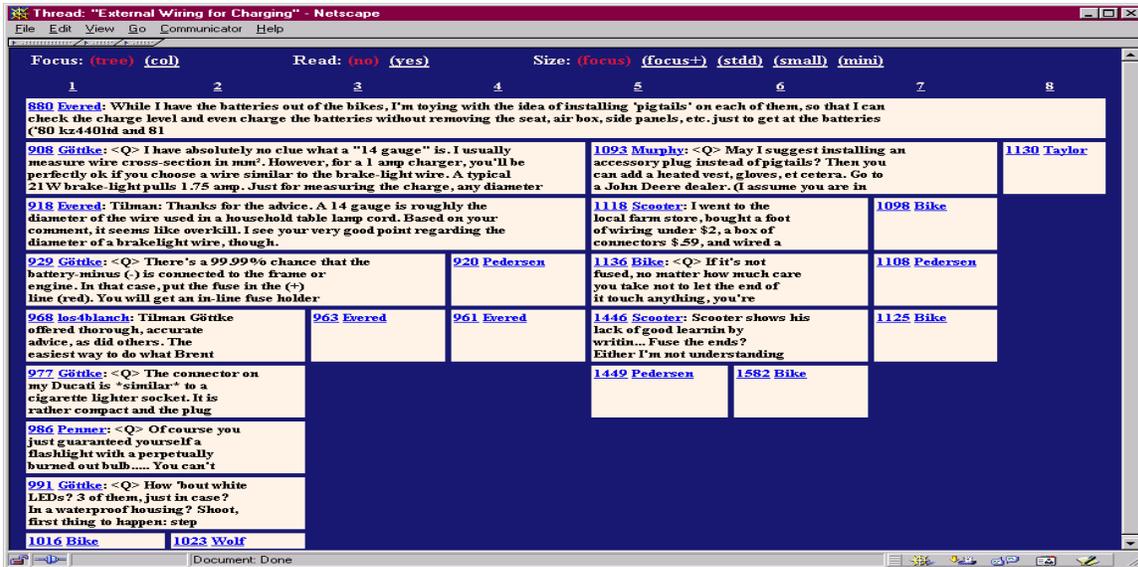


Figure 6. Treetable

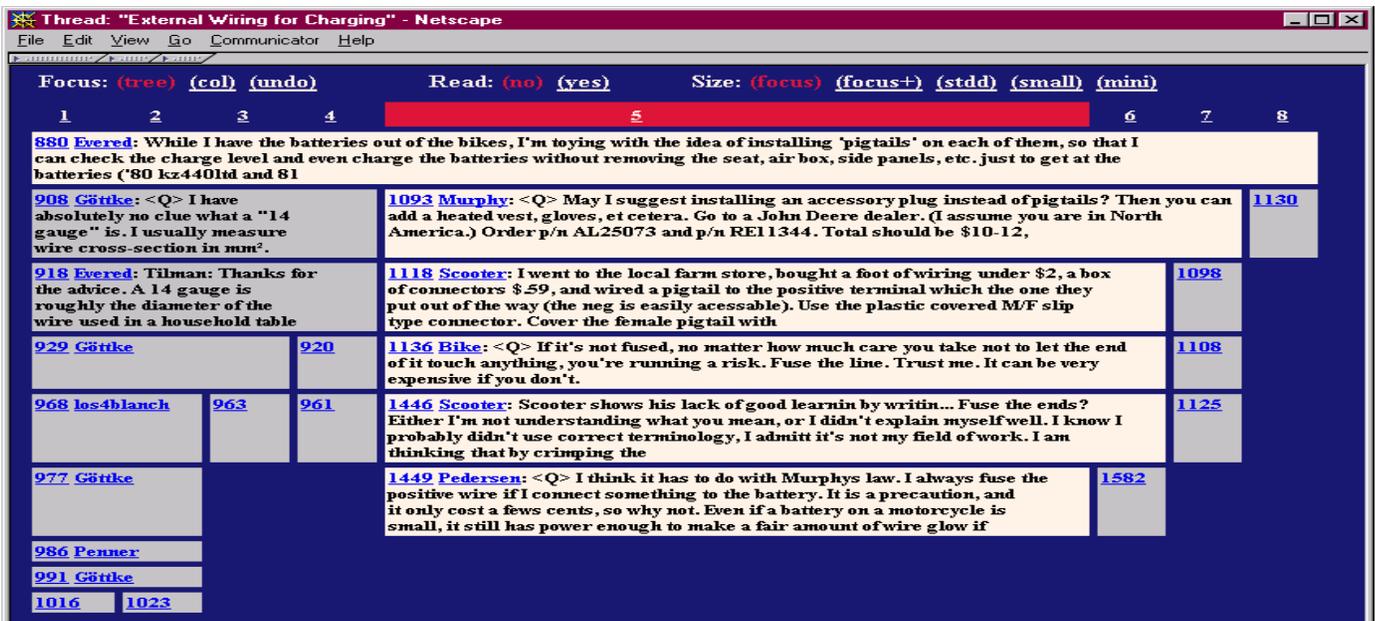


Figure 7. Treetable with focus on column 5

The adaptation to treetables allows (by modal switch or column selection) either subtrees or individual columns to be selected as foci, and to be expanded to different degrees. In figure 7, column 5 is selected. The selection causes the column to be highlighted, and expanded to allow more room for text for cells spanning that column

For full reading of a thread, treetables are used as guides to auxiliary displays presenting the full essential text of either a column, as shown in figure 8, or the immediate responses to a message, as shown in figure 9. In both cases, the selected element on the treetable guide is highlighted to maintain context.

In the case of column displays, the result in many cases does read like a theatrical script, and quotes representing the entire content of an immediately preceding message can be removed completely. Also, references back to the guide treetable to view additional sub-conversations are needed only at the ends of the columns, and the references seem more convenient because the structure is clearer than in the case of narrow-tree “overview + detail” presentations.

Several additional treetable variations have been developed. For very large treetables, more comprehensible subtrees can be extracted into additional windows. Also, if display space is limited, the “guide” treetables can be reduced to outline form to fit into minimum size windows, or a framed approach can be used. Space precludes illustration of any but the latter. A framed view is illustrated in figure 10. In this version, a vertically partitioned outline tree treetable is used as a guide to reading the right-hand frame, both to maintain context and, by selection, to navigate among columns.

MESSAGE AND THREAD ANALYSIS

Underlying the displays described in the preceding sections is a detailed analysis of each message in an archive as it arrives, followed by (incremental) threading. The process of analyzing a message is described in the next sub-section, followed by a brief description of the approach used in threading.

Analyzing messages

As discussed in earlier sections, the results of message analysis are used to find essential texts. They are also used to find substantive fragments for thread overviews.

The result of message analysis is a message-component tree with individual lines at the leaves. The upper levels of the tree separate the message into the main body and (recursively) nested excerpts, and then separate each such subdivision into a type-specific block, e.g., a prose paragraph or a signature block.

The analysis underlying the tree cannot be done by a traditional deterministic parse, because message structures do not conform to any fixed grammar. Instead, the analysis is based on encoding a constantly evolving set of

conventions, and using them to obtain a best-guess partitioning of the message.

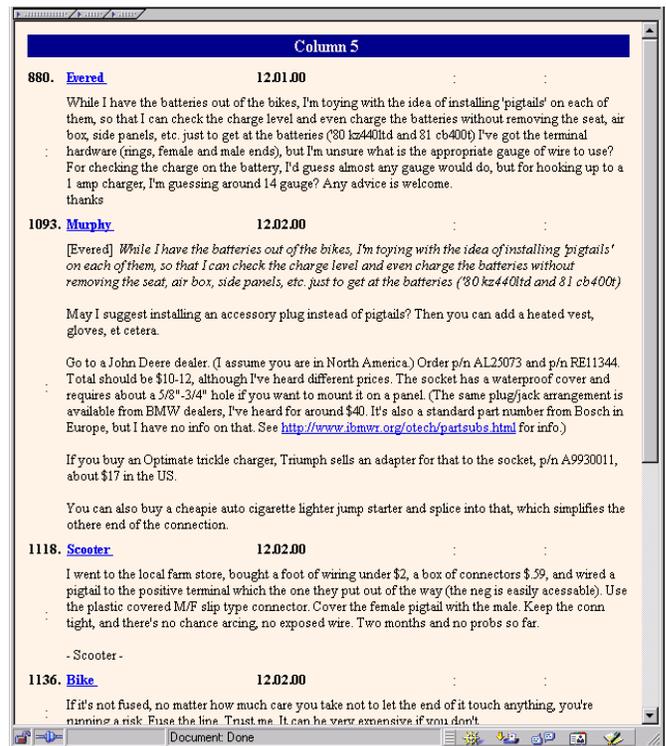


Figure 8. Concatenated message texts for column

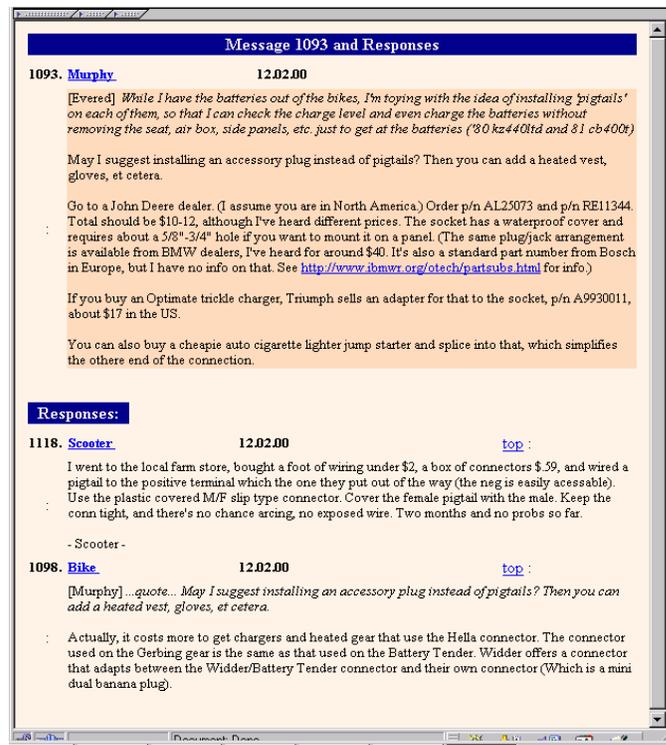


Figure 9. Texts for message and responses

The analysis is divided into three stages. First, a recursive descent process is used to delineate headers, nested excerpts and their accompanying introductory material. Then the main body and excerpts are further analyzed, separately, using a kind of weighted finite state transducer. In the final stage, the trees obtained by analyses of excerpts are inserted at the appropriate places in the tree representing the main body of the message.

The two analytic stages could be collapsed into a single finite-state-like process. But the separation into recursive descent and finite-state stages, while possibly less elegant, significantly simplifies the finite-state aspect.

The weighted finite-state-transducer is not a conventional one. Arc input labels are not input symbols but, rather, the names of procedural tests, such as "test for prose", that are applied to message lines and that return goodness values. Arc output labels are sequences of message tree node types, for example "begin-paragraph+prose-line". The output network, obtained by running the transducer on the message, contains one or more arcs for each line, each associated with an arc output label and a test result value. The message component tree is constructed from the output network by finding the maximal path through the network (a linear procedure), and then submitting the labels on that path to a tree constructor. For example, the sequence "begin-paragraph+prose-line" would cause the creation of a new paragraph block with a prose line as its first child.

Related message analysis work

The finite-state-like approach, at least in the use of line-specific procedural tests with "goodness" values as results,

is closely related to one used by Chen et.al. [9] to analyze the final portions of messages, which present the largest challenges, and especially signature blocks, for a text-to-speech application. Because of the different purpose, the signature block analysis in [9] is considerably more intensive.

Thread Analysis

Delineating thread structure consists of identifying, for each message, a "best" thread predecessor. While it is acknowledged that a particular message may in fact represent a response to several prior messages, we want to represent the thread, to the extent possible, in the form of readable message/response chains.

The identification of thread structure is accomplished by combining evidence from such things as explicit header fields (e.g., "in-reply-to", "references"), the headers and introductions of excerpted messages, and the tracing of quoted passages to their sources. This has been done many times before, at varying levels of detail.

The novel element here is that quoted prose passages are related to their sources by matching sentences rather than lines, as is done, for example, by Tajima et.al [10]. This is useful both because quotes often select sentences rather than lines, and also because some mailers may change the line partitioning of quoted passages. Sentence matching is based on hashing the initial 20 characters of the sentence and, for longer sentences, the final 20 characters as well. The latter is useful for elided quotes that may, for example, consist of the initial part of one sentence, then an elision marker (...), and then the final part of another sentence.

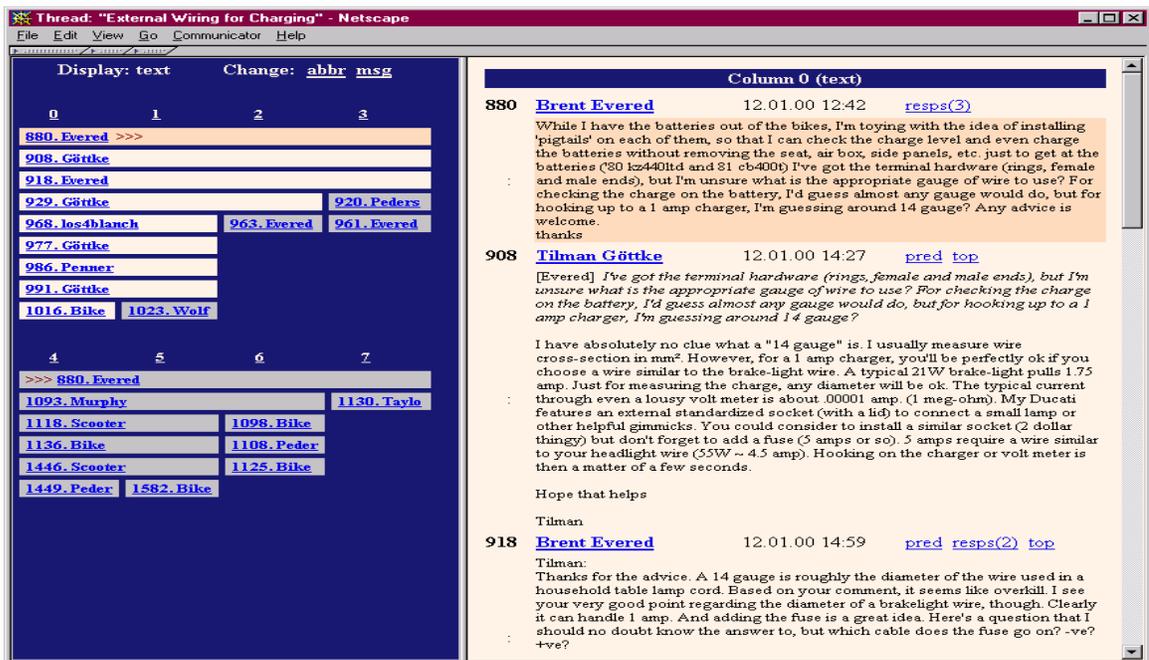


Figure 10. Framed treetable reading view

INITIAL REACTIONS

While no formal user tests have been conducted on the MailContent facilities, they have been in use for a considerable time as part of a local portal that lists (among other things) the more recent threads in organization-wide distribution lists. They have also been the subject of many demonstrations and discussions.

Two significant reactions to the facilities should be considered in directing future work. The first, and fairly frequent, reaction concerns the inattention to the social elements of archive portrayal, which are major concerns of, for example, the work of Sack [11] and Smith and Fiore [6]. As indicated by reactions reported in [6], both content and social elements must be combined. This can be done rather easily in the MailContent framework. For example, adapting an approach of [6], treetables can be augmented by lists of contributors, and then selection of a contributor might highlight their contributions.

A second reaction concerns a difficulty in reorientation in shifting between expansions of different focus areas in wider treetables, even though the regular structure is preserved. While perspective can be regained by intermediate shifts between focused and non-focused forms, animation of the focus shifts may be useful in dealing with this problem, requiring facilities beyond those of the conventional browser currently used for the displays.

CONCLUDING REMARKS

This paper describes the thread-level facilities of an existing, integrated system for email archive exploration. The facilities combine analysis and presentation techniques to provide useful overviews of thread content, and to expedite the reading and assimilation of subject matter of interest. We consider treetables to be the more promising of the two new visualizations for complex threads, because of the clarity of the structural representation and because the associated column reading views so frequently do achieve play-like reading convenience. However, narrow trees seem to retain some advantages for content overviews and for reading shorter threads, and, therefore, both visualizations have been retained in the system as alternatives.

Our future work in the area should include improvements dictated by the initial reactions described in the preceding section, and refinements grounded in task-oriented studies. Additional directions of considerable interest include real thread summarization, which remains a tantalizing, albeit elusive, goal, and the development of automated means of adapting message analyses to different email corpora and mailing list conventions. McCallum et.al. [12] describe a training-based method for partitioning messages within an FAQ digest into four elements: header, question, answer,

and "tail". The training uses features such as "contains-question-mark" to learn a partitioner from a set of training documents labeled by line type. While it remains to be seen whether this method can be adapted to more general message types and components, the automation is certainly of interest in the context of corpus specific, and ever changing, message structure conventions.

REFERENCES

1. Hypermail development archive. Available at <http://www.hypermail.org>
2. Jackson, S., Yankelovich, N., Intermail: a Prototype Hypermedia Mail System. *Proceedings of Hypertext '91* (San Antonio TX, Dec. 1991) 405-409
3. Google groups. Available at <http://groups.google.com/>
4. Plaisant, C., Carr, D, Schneiderman, B., Image-Browser Taxonomy and Guidelines for Designers, *IEEE Software* 12,2 (March 1995) 21-32
5. Yabe, J., Takahashi, S., Shibayama, E., Automatic Animation of Discussions in USENET. *Proceedings of AVI 2000* (Palermo, Italy, May 2000) 84-91.
6. Smith, T., Fiore, A., Visualization Components for Persistent Conversations. *Proceedings of CHI '01* (Seattle WA, May 2001), ACM Press, 136-143
7. Popolov, D., Callaghan, M., Luker, P., Conversation Space: Visualizing Multi-Threaded Conversation. *Proceedings of AVI 2000* (Palermo, Italy, May 2000) 246-249.
8. Rao, R., Card, S.K., The Table Lens: Merging Graphical and Symbolic Information in an Interactive Focus + Context Visualization for Tabular Information. *Proceedings of CHI '94* (Boston MA, April 1994) 318-322
9. Chen, H., Hu, J., Sproat, R.W., Integrating Geometrical and Linguistic Analysis for Email Signature Parsing. *ACM Transactions on Information Systems* 17,4 (Oct. 1999) 343-366
10. Tajima, K., Mizuuchi, Y., Kitagawa, M, Tanaka, K., Cut as a Querying Unit for WWW, Netnews, E-mail. *Proceedings of Hypertext '98* (Pittsburgh PA, June 1998) 235-244
11. Sack, W., Conversation Map: a Content-based Usenet Newsgroup Browser. *Proceedings of IUI '00* (New Orleans LA, January 2000) 233 – 240
12. McCallum, A., Freitag, D., Pereira, F., Maximum Entropy Markov Models for Information Extraction and Segmentation. *Proceedings of ICML '00* (Stanford CA, June 2000) 591-598