Summarizing Archived Discussions: A Beginning

Paula S. Newman
Palo Alto Research Center (PARC)
3333 Coyote Hill Road
Palo Alto, CA, 94304 USA
+1 650- 812- 4239
pnewman@parc.com

John C. Blitzer
University of Pennsylvania
Moore School Building
200 South 33rd Street
Philadelphia, PA 19104 USA
blitzer@cis.upenn.edu

ABSTRACT
This paper describes an approach to digesting threads of archived discussion lists by clustering messages into approximate topical groups, and then extracting shorter overviews, and longer summaries for each group.

Categories and Subject Descriptors
H.3.1 [**Information Systems**]: Content Analysis and Indexing -- *Abstracting Methods*; H.3.3 [**Information Systems**]: Information Search and Retrieval – *Clustering*; H.4.3 [**Information Systems Applications**]: Communications Applications -- *bulletin boards*

General Terms
Algorithms, Human Factors, Experimentation

Keywords
Discussion lists, newsgroups, digests, clustering, summarization, persistent conversations.

1. INTRODUCTION
The proliferation of highly active newsgroups and mailing lists has led to vast collections of archived conversations that are a potentially valuable resource. One way of helping readers select conversations (threads) of interest is via thread pseudo-digests that embed substantive initial fragments of each message within new types of tree representations [5]. However, this approach fails for the very large, often multi-subtopic threads most in need of abbreviation, because the combined size of the fragments is still very large. In this paper, we describe methods for digesting long stored conversations by (a) clustering messages into subtopic groups and then (b) forming relatively short overviews or longer summaries for each group. Figures 1 and 2 illustrate some results obtained by the implementation. Figure 1 shows overviews of two clusters of a 93-message thread of the rec.motorcycles newsgroup ostensibly concerning BSA motorcycles, but devoting considerable space to a discussion of purchasing ethics. Figure 2 shows a longer summary of one of those clusters.

IUI'03, January 12–15, 2003, Miami, Florida, USA.
ACM 1-58113-586-6/03/0001.

2. MESSAGE CLUSTERING
Messages are clustered into subtopic groups by forming word vectors for each message and then using a form of nearest neighbor clustering. An important aspect of word vector formation is the adjustment of messages to avoid distortions in lexical distance due to differences in quoting style. The method used rests on the observation that responses to longer messages usually selectively quote the issues of concern, unless the responses are digressions or meta-comments, while short responses to short messages usually focus on the entire subject of the parent message. Therefore, if a message contains selective quotes, we do not include a non-selective quote of the parent in the adjusted response. However, if there are no selective quotes, we include the parent in the response if both messages are relatively short (less than 350 characters counting only non-stop words) and not otherwise.

After messages are adjusted, we build an inverse-document-frequency (idf) [8] weighted word vector for the stemmed, non-stop words of each message. To further decrease the distance between logically related messages, we then apply probabilistic latent semantic analysis (PLSA) [3,1]. The result form used characterizes each message as a vector of word probabilities, with some words not contained in the message having significant probabilities if they occur with contained words in other contexts.

2.1 Clustering Algorithm
The clustering approach considers only parent-child and sibling message pairs of the thread tree. Messages are first placed in individual clusters, and then grouped by a single-link agglomerative clustering process [8]. At each step, the two clusters linked by the most similar message pair are combined. The process halts when either (a) the combined size of the two clusters is greater than a size threshold T, or (b) the distance between the messages in the linking message pair is greater than a distance threshold D. If the process halts because of size thresholding, the two clusters involved are combined if the resulting size is less than a given value M > T. During clustering, the root message of the thread is held out, and later combined with the closest cluster, to separate subtopics dealing with different issues raised in the root message.

The size threshold T is currently set, based on experimentation, to 1/3 of the number of messages in the thread, but not less than 25 nor more than 40 messages. The distance threshold D is currently 75% of the largest distance

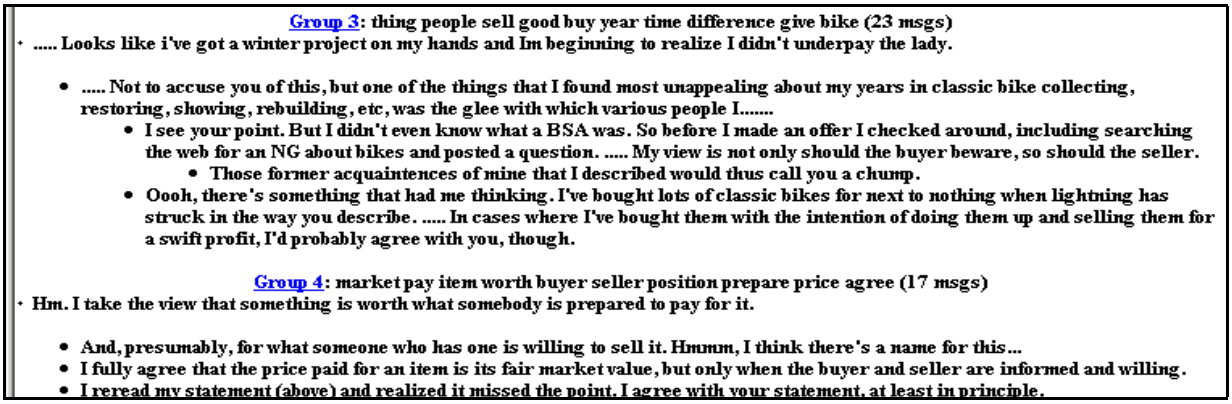

Figure 1. Overviews of two subtopic clusters (groups) of a thread

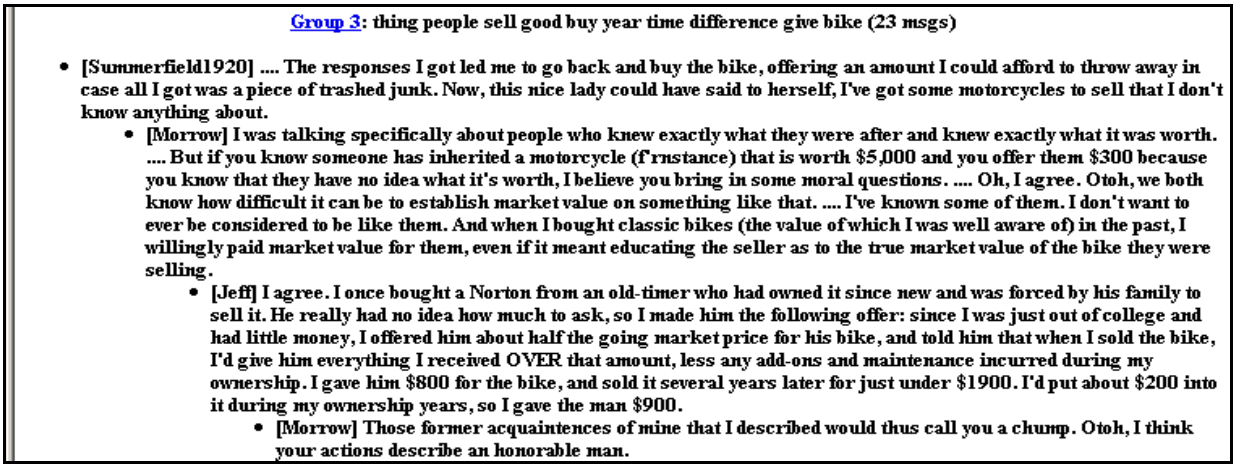

Figure 2. Summary of group 3 of figure 1

of the parent-child and sibling pairs. After clustering halts, final clusters larger than a given minimum size are selected for use in the digest.

2.2 Initial Clustering Results

The size threshold T is usually reached before the distance threshold D, and is more effective as a halting criterion than any absolute or distribution-based distance threshold we have explored. It is consistent with the observation that in larger discussions containing distinguishable subtopics, the size of at least one such subtopic is usually proportional to the size of the thread. In an initial test of the method, using a sample of 29 threads of 50-162 messages in length from the rec.motorcycles newsgroup containing subtopics, we find about 87 relatively important subtopics. Of these, 72 are represented either by single clusters (54), or are still easily identified although divided into 2 clusters (18). The remaining 15 subtopics are split among more than 2 clusters, or are not represented by clusters.

2.3 Related Clustering Work

Because an email thread is not a collection of independent documents, but an evolving conversation, the clustering method used is related to methods for document segmentation [1,2] that are concerned with distances between consecutive elements.

Two efforts deal directly with message clustering within threads. Tajima et. al. [9] identify overlapping subtrees of a thread as units of retrieval. They process the thread tree bottom-up and, at each step, combine a node with its currently open child subtrees, separately or together, if the similarity between the node word vector and the centroid vector of the child subtree or subtrees exceeds an (unspecified) absolute threshold. No results are given, but if the threshold is set high, it is likely that the method will obtain shallow subtrees suitable as query results.

Also, Ozaku et.al. [6] retrieve threads relating to a topic, and then filter subtrees not dealing with that topic. They use noun keywords to represent messages, and try to find "topic changing articles" where the proportion of never seen keywords shifts, and "topic branching articles" whose responses differ in keyword usage and quoted passages

3. OVERVIEW FORMATION

Different methods are used for developing overviews, and more lengthy summaries, of the clusters to be represented in the digests. Overview development is described in this section, and summary development in the next one.

The overview for a cluster *c* is formed by selecting a set of messages, and then one or more sentences from each such message, with the goal of obtaining semantically central quote/response sequences.

The set of messages ***M*** from which overview sentences will be selected is made up of a core set *CM,* and an auxiliary set ***AM***. ***CM*** is initialized with a given proportion of messages closest to the overall lexical centroid for c. These messages will be lexically central based on both their quoted and non-quoted content. ***CM*** may be extended in two ways. First, if *c* contains the thread root message and/or its successors, the root and a proportion of successors may be added to ***CM*** to provide additional coverage of messages likely to be of most interest to the casual reader. Second, for broader coverage, ***CM*** may be extended by some messages that were central to some sizeable clusters found during the clustering process and later combined with other clusters. After these extensions, if any, *CM* is pruned to a predetermined proportion of the cluster size.

The auxiliary set ***AM*** contains messages not in ***CM*** that are predecessors of messages in *CM*, or have more than 2 responses in *c*. It is used as the source of quoted passages. Figure 3 shows a possible set of core messages, shown in solid-outline rectangles, and auxiliary messages, shown in ovals, for a cluster. The sentences in each message are indicated abstractly. Thus message m0 contains sentences S0, S1, and S2, while message m4 contains sentences S7, QS2 (quoting S2), and S8.

After forming ***M***, quoting relationships used to create extracted quote/response sequences are found. For each message ***m*** in ***M*** a set of quoted sentences ***Qm*** is identified, containing sentences in *m* quoted by responses to *m*.

The sentences in ***Qm*** are those deemed most important; currently, these are the last quoted sentences immediately preceding longer sequences of response material. For Figure 5, the non-null sets *Qm* would be *Q0* ={S2} and *Q4* ={S8}

Then, for each message *m* in ***M***, the string that will represent *m* in the summary is compiled, as follows:

a) If *m* is the thread root, include a long initial substring.

b) If ***m*** is in ***CM*** or ***Qm*** is null, include the sentence of *m* beginning the longest passage of *m* following a quote in *Qp*, where *p* is the parent of *m.* If there is no such sentence, then if ***m*** is in ***CM*** include the initial sentence of m, otherwise the final sentence

c) If *Qm* is non-null, include the sentences in *Qm*.

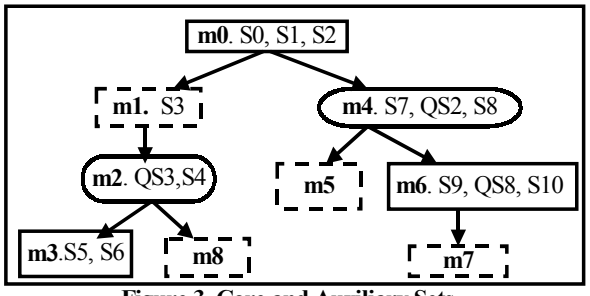

Figure 3. Core and Auxiliary Sets

Table 1 illustrates the sentences that would be selected from the cluster of Figure 3. As mentioned earlier, figure 1 illustrates overviews of two clusters about purchasing ethics from 93-message thread ostensibly about BSA motorcycles. Each overview is headed by a list of the most frequent (idf-weighted) words in the cluster to further characterize the subtopic, and is linked to a region of a two dimensional representation of the full thread tree [5].

4. SUMMARY FORMATION

The approach to summarization builds on standard methods of feature-based summarization [4,7,10], adding techniques and features that are specific to discussion list threads. The summaries are adjustable as to length. In our work we use a traditional length of 10% of the number of sentences in the source, or 60% of the number of messages, whichever is larger. The resulting summaries are substantially longer, and more informative than the above overviews. Figure 2 shows the summary of group 3 of Figure 1.

The process of summarization for a cluster begins by computing an intrinsic score for each contained sentence, based on its position in the cluster and its lexical centrality. The process then iterates over the cluster until the summary reaches the desired size. At each iteration it computes a new extract score for each sentence not yet in the summary, and adds the sentence with the best such score to the summary. The extract score estimates the potential contribution of the sentence to summary coherence and coverage. The score computations are given below.

4.1 Computing Intrinsic Scores

The intrinsic score *Iscore(s)* for a sentence *s*, is computed as

$$\alpha * P(s) + (1-\alpha) * S(s), \quad 0 \le \alpha \le 1$$

The position score *P(s)* is computed as

$$\beta \cdot W(s) + (1-\beta) \cdot \sqrt{1/pos(s)}, \quad 0 \le \beta \le 1$$

pos(s) is the ordinal position of *s* in its containing message. *W*(s) measures the importance of the message containing *s*, computed as $\log(w(s))/\log(wmax)$ where *w(s)* is the number of nodes in the subtree rooted at the message containing sentence *s* and *wmax* is the maximum value of *w(s)* for all sentences in the cluster.

The lexical similarity *S(s*) of the sentence to the cluster centroid is computed using a word pseudocount vector $\vec{s}$, which can be simply a vector of tf.idf weighted, stemmed, stopped word counts. Alternatively, a PLSA model [1,3] can be trained across a corpus of approximately 1000 messages,

Table 1. Extracted Sentences

m0	**S0** (*Qp* null so use first), …**S2** (from *Q0*)
m2	**S4** (*Q2* null, so use last)
m3	**S5** (first)
m4	**S8** (from *Q4*)
m6	**S10** (after quote in *Q4*)

and then $\vec{s}$ computed by folding the word vector for *s* into the PLSA model, obtaining a vector that describes a probability distribution over latent classes *z* for sentence *s*:

$$\vec{s} = \left\langle p(z_1 \mid s), p(z_2 \mid s), ..., p(z_n \mid s) \right\rangle$$

S(s) is then the Hellinger similarity of $\vec{s}$ to the lexical centroid $\vec{c}$ of the sentences in cluster $\vec{c}$:

$$S(s) = \sum_{j=1}^{n} \sqrt{\vec{c}^{\,j} \cdot \vec{s}^{\,j}}$$ where $\vec{s}^{\,j}$ indicates the jth component of the vector $\vec{s}$.

4.2 Computing Extract Scores

At each iteration, an extract feature score *EF(E,s)* is computed for each sentence *s* and combined with its intrinsic *IScore(s)* to yield an overall extract score

$$EScore(E,s) = IScore(s) + EF(E,s) \bullet IScore(s)$$

and the sentence with the best *EScore* is added to the extract. The extract feature score *EF(E,s)* is a weighted sum of feature values

$$EF(E,s) = \gamma_1 \cdot Adj(E,s) + \gamma_2 \cdot TreeAdj(E,s) + \gamma_3 \cdot Quote(E,S) - \gamma_4 \cdot LexSim(E,s)$$

measuring the relationship of *s* to sentences already in the extract. The features used in *EF(E,s)* are:

1. Adjacency *Adj(E,s)* to sentences already in the abstract counts the sentences in the extract to which s is adjacent, with a maximum of 2.
2. Tree adjacency *TreeAdj(E,s)* to messages already in the abstract is computed as:

   $$\sum_{s_e \in E} ctest(s_e, s)$$ where *ctest (s1, s2)* is 1 if the message containing *s1* responds to the message containing *s2*, or vice versa, and 0 otherwise.
3. Quote relationship *Quote (E, s)* is measured by

   $$\sum_{s_e \in E} qtest(s_e, s)$$ where *qtest(s1, s2)* is 1 if *s1* is immediately preceded by text that is marked as a quote and is identical to *s2*, or vice versa, and 0 otherwise.
4. Lexical Similarity. *LexSim(E,s)* is calculated by summing the Hellinger similarity between *s* and each sentence in the extract. It is negatively weighted in the extract feature score, and used to reduce redundancy in the summary.

Building summaries is fairly resource-intensive because of the iterative extract-building process. It becomes much more resource-intensive, and somewhat more effective, if the word pseudocount vectors are based on large-scale PLSA computations.

5. CONCLUDING REMARKS

The methods described above have been applied to a number of discussion lists with encouraging results. However, far more work is needed in measuring and tuning the methods, in experimenting with deeper analyses for quote normalization and clustering, and in evaluating the utility of the digests to users.

6. ACKNOWLEDGEMENTS

We thank Francine Chen for suggesting the use of nearest-neighbor clustering, Thorsten Brants for advice on use of his PLSA implementation, and Thorsten Brants and Annie Zaenen for their helpful comments on drafts of this paper.

7. REFERENCES

[1] Brants, T., Chen, F., and Farahat, A. Arabic document topic analysis, LREC-2002 Workshop on Arabic Language Resources and Evaluation, (Las Palmas, Spain, June 2002)

[2] Hearst, M. Segmenting text into multi-paragraph subtopic passages. Computational Linguistics 23, 1 (March 1997) 33-64

[3] Hofmann, T. Probabilistic latent semantic indexing, in Proceedings of SIGIR '99 (Berkeley, CA, August 1999) 50-57.

[4] Kupiec, J., Pedersen, J. O., and Chen, F. A trainable document summarizer. in Mani, I and Maybury, M.T. (eds.) Research and Development in Information Retrieval. MIT Press. Cambridge MA, 1995, 55-60

[5] Newman, P. Exploring Discussion List Archives: Steps and Directions, in Proceedings of JCDL '02 (Portland, OR, July 2002) 126-134

[6] Ozaku, H., Uchimoto, K., Murata, M. and Isahara, H. Topic search for intelligent network news reader HISHO. in Proceedings of ACM SAC 2000 (Como, Italy, March 2000) 28-33

[7] Radev, D., Jing, H., and Budzikowska, M. Centroid-based summarization of multiple documents. in ANLP/NAACL '00 Workshop on Automatic Summarization (Seattle, WA, April 2000) 21-29

[8] Salton, G. Automatic Text Processing. Addison Wesley (1989) 280, 329

[9] Tajima, K., Mizuuchi, Y., Kitagawa, M., and Tanaka, K. Cut as a querying unit for WWW, Netnews, and E-mail, in Proceedings of ACM Hypertext '98 (Pittsburgh, PA, June 1998) 235-244

[10] White, M., and Cardie, C. Selecting sentences for multi-document summaries using randomized local search, in Proceedings of ACL '02 Summarization Workshop (Philadelphia, PA, July 2002