Running Head: THEME MACHINE THEORY AND METHODS

The Theme Machine: Theoretical Foundation and Summary of Methods

Bruce L. Lambert

University of Illinois at Chicago

Author Note

Bruce L. Lambert (Ph.D., University of Illinois at Urbana Champaign, 1992) is Assistant Professor of Pharmacy Administration and Clinical Assistant Professor of Pharmacy Practice at the University of Illinois at Chicago. The author is grateful to William Frakes for making his stemming and stopping algorithms available and to David Lambert and the members of the comp.lang.lisp Usenet newsgroup for programming assistance. Thanks also to the members of the University of Illinois Working Group on Message Design, especially Joe Bonito, for their willingness to experiment with prototype versions of the Theme Machine. This paper was prepared for presentation at the 46th Annual Conference of the International Communication Association, May 25, 1996, Chicago, Illinois.

Address:	833 S. Wood St. (M/C 871), Chicago, IL 60612-7231
Phone:	(312) 996-2411
Fax:	(312) 996-3272
Internet:	lambertb@uic.edu, bruce@ludwig.pmad.uic.edu
Anonymous FTP:	ludwig.pmad.uic.edu

Abstract

A theory of message design attempts to explain the relationship between message structure and message function. In order to formulate and investigate questions about message design, one must have an independent representation of message structure and message function. Experience has shown that it is possible to develop content analytic coding systems that are based primarily on patterns in the structure of discourse, but it is extremely labor intensive to do so, and the resulting coding systems often have many categories and are difficult to reliably apply. In an effort to overcome these methodological problems, an automated coding system, dubbed the Theme Machine, has been developed. The Theme Machine does automatic content analysis via document clustering. The techniques underlying the Theme Machine were originally developed to enhance information retrieval effectiveness and efficiency. They apparently have not been used previously for the purpose of content analysis. This essay describes the theoretical rationale for the development of the Theme Machine and summarizes the methods used to implement the Theme Machine. Preliminary investigations in a variety of contexts indicate that the methods embodied by the Theme Machine may have broad utility among communication researchers.

The Theme Machine: Theoretical Foundation and Summary of Methods

This paper describes an automated system, developed by the author, for extracting thematic features from computer-readable text. The system is called the Theme Machine. The system is based on methods for document clustering that were originally developed to facilitate information retrieval (Griffiths, Robinson & Willett, 1984; Jardine & van Rijsbergen, 1971; Rasmussen, 1992; Salton, Allan, Buckley & Singhal, 1994; Salton & McGill, 1983; Voorhees, 1986; Willett, 1980; Willett, 1983; Willett, 1988). These methods have apparently not been used before for the purpose of content analysis (Vorhees, personal communication; Willett, personal communication). Preliminary investigations in a variety of contexts indicate that the methods may have broad utility for communication researchers (Bonito, 1996; Goldsmith, Dun, MacGeorge & McDermott, 1996; Lambert, 1996; O'Keefe, 1996). In order to encourage the wider application and development of similar methods and techniques, this essay offers both a theoretical rationale and a methodological primer on the procedures used by the Theme Machine.

Overview

The first section gives a brief, high-level description of the Theme Machine’s structure and function. The second section puts forward a theoretical rationale for automatic coding, emphasizing the need to have an account of message structure that is fully independent of message function. The third section delves into the inner workings of the Theme Machine, touching on the transcription of discourse into computer-readable text, the segmentation of whole texts into units, the removal of common words and suffixes, the representation of documents as vectors of term weights, and the extraction of thematic features via document clustering. Section four describes how the output of the Theme Machine might be used to analyze the relationship between message structures and message effects. The final section summarizes the main points and discusses directions for future research.

Brief Description of the Theme Machine

The Theme Machine (TM) is a set of computer programs written primarily in the programming language LISP. The TM takes as input a plain (ASCII) text file containing an arbitrary number of discourse units and produces as output a set of clusters whose members are thematically-related units. A similarity threshold influences the number and quality of the resulting clusters. Adopting the vocabulary of information retrieval researchers, each word is called a _term_. Each discourse unit is referred to as a _document_, and the entire set of documents is known as the _collection_. The TM begins by preprocessing each document in the collection. The preprocessing involves the removal of common terms and the reduction, by suffix stripping, of remaining terms to their stem forms. The frequency of occurrence of each unique term in the collection is tallied, and each document is represented as a vector of appropriately weighted term frequencies. Document vectors are then clustered by standard hierarchical agglomerative methods. To review, the TM takes segmented, computer-readable text and identifies content analytic categories (i.e., themes) via document clustering. Categorical variables are created to represent the presence/absence of each theme. Messages (discourse units) are represented in terms of these categorical variables. Goals and message effects are then analyzed in relation to these variables. Having explained very briefly _what_ the TM does, the next section explains _why_ one might want to use a computer to automate certain stages in the process of content analysis.

Theoretical Foundations for the Theme Machine

In a series of articles, O’Keefe and Lambert have argued for a local management model of message design (for a review, see O'Keefe & Lambert, 1995). A thorough description of the model is beyond the scope of this essay. Only a summary is given here, highlighting the points most relevant to the use of the TM. According to O’Keefe and Lambert (1995), a theory of message design is understood as

> a systematic theory of the relationship between message structure and message function. _Message structure_ refers to the substance, organization, and placement of discourse. _Message function_ involves both the antecedent conditions of message generation (especially the goals of the message producer) and the intended and unintended consequences of the message. (O'Keefe & Lambert, 1995, p. 54)

To date approaches to message design have largely been holistic and functional. Discourse units (typically whole messages) are identified and then functionally categorized, and the functional categories (e.g., lists of strategies) are analyzed in relation to situational features, goals, message effects, and so on. This approach stimulated a great deal of research, but its limitations are now increasingly being recognized (Kellerman & Cole, 1994).

The Conflation of Message Structure and Message Function

One problem with holistic functional analyses is that they conflate message structure and message function. Perhaps the best example of the conflation of message structure and function comes from plan-based models of message production (Appelt, 1985; Cohen & Perrault, 1979). Such models assume message production is a two-step process: first reason from goals to acts (i.e., strategies, speech acts), then instantiate acts as concrete utterances (i.e., words, sentences). So for example, if you have the goal of finding out whether John is here, and you believe that making a request should satisfy this goal, then you attempt to formulate a request. When one tries to formulate the request, one confronts what O’Keefe & Lambert (1995) call the instantiation problem. That is, how does one get from---make a request about John’s being here---to “Has anyone seen John?”

Most planning models solve this problem through the use of a functional indexing scheme, and it is in the functional indexing scheme where message structure and function are conflated. A functional indexing scheme is a system for representing, storing and retrieving linguistic structures. In a functional indexing scheme, linguistic structures are indexed by their putative functions. In the example given, the speech act of requesting is associated with the interrogative sentence type (Cohen & Perrault, 1979). In related models, words and phrases are associated with the achievement of certain rhetorical goals (Hovy, 1990). Even at low levels of instantiation, morphemes, phonemes, and intonation patterns are associated with specific functional significance, with the achievement of particular goals (Levelt, 1989). With a functionally indexed lexicon of linguistic forms one can search by specifying the goal to be achieved and thereby retrieve a linguistic form that will (supposedly) achieve that goal.

When the representation of a linguistic form includes a specification of the form’s functional significance or a description of the goals the form will help to achieve when it is expressed, form and function are hopelessly conflated. Assigning functional significance to decontextualized linguistic forms ignores indirectness and the pervasive context-sensitivity of meaning. Worse still, when form and function are conflated, it becomes impossible to ask questions about the relationship between message form and message function. In order to construct a theory of message design, a theory that investigates the relationship between message structure and message function, <u>one must have independent representations of message structure and message</u>

function. Most content-analytic coding schemes used in contemporary communication research are typologies that conflate structure and function.

Creating An Independent Representation of Message Structure

This brings us to the potential utility of the TM or any other automated system for content analytic coding. The TM was developed in an effort to create a representation of message structure (what we sometimes call content) that was independent of message function. Initially, we reasoned that a “function free” set of categories should be based only on patterns in the literal content, the surface structure, of discourse units. Units would be grouped together only if they contained similar (often identical) patterns of words or phrases (Lambert, 1992; Lambert, 1995; Lambert, 1996; Lambert, in press; Lambert & Gillespie, 1994; Saeki & O’Keefe, 1994). Although this approach proved useful in illustrating the relationship between message structure and function in a variety of contexts, it was beset by difficult methodological problems. The main problems involved the efficient creation and reliable application of categorical coding systems based only on patterns in the literal content of discourse units. These coding systems were initially developed manually and inductively. The process often took several weeks or even months. The result was a set of structural categories. The number of categories was often large, ranging from a low of 22 to as many as 70 or more.

The sheer number of categories created two difficulties. First, it was difficult to achieve a satisfactory level of interrater reliability with so many categories to choose from, and standard methods for assessing interrater reliability were not well suited to category systems with high numbers of categories, where the frequency of occurrence of some categories is relatively low (Cohen, 1960). The second problem was that the large number of categories made analysis of message-effect relations cumbersome and sometimes impractical. In practice, the dimensionality of the original coding schemes was reduced prior to statistical analysis by grouping low level categories (referred to as idea-types) in to higher level categories (referred to as themes) (see e.g., Lambert & Gillespie, 1994; Saeki & O'Keefe, 1994).

The Theme Machine as a Solution to Message Coding Problems

The TM was designed to address the methodological problems discussed above. The goal was to develop a system that could automatically and deterministically cluster documents based on the presence of similar patterns of words and phrases. By automating the system, the efficiency problems could be overcome, and by making the system deterministic, the problem of interrater reliability could be sidestepped completely. (A deterministic system has no random element. Given the same initial conditions, it will always proceed through the same sequence of actions and arrive at the same end point.) Although still a prototype, the TM, as currently implemented, appears to have successfully achieved the design goals that were set out for it. In the following sections, the details of the TM algorithm and implementation are discussed.

The Theme Machine: Methods

As noted above, analyzing data with the TM involves a sequence of several steps. These steps are as follows: (a) transcription, (b) unitization, (c) tabulation of term frequencies, (d) removal of common terms, (e) reduction of terms to stem form, (e) assignment of term weights, (f) creation of an inverted file, (g) creation of an interdocument similarity matrix, and (h) clustering of documents. Each of these steps is described in turn.

Transcription and Unitization

The first two steps in the sequence, transcription and unitization, are not yet automated, but they are nonetheless crucial. The discourse to be analyzed can originally be oral or written, but to be input to the TM it must be transcribed into plain (ASCII) text. Once transcribed, the discourse must be segmented into units. The units can be defined at any grain size the analyst chooses (e.g., phrase, clause, sentence, turn, paragraph, chapter, etc.). Members of the University of Illinois Working Group on Message Design have typically selected the independent clause as the unit of analysis (Lambert & Gillespie, 1994; Saeki & O'Keefe, 1994). Evidence from both oral and written discourse suggests that the independent clause can be construed as the smallest linguistic unit that expresses a complete thought (Chafe, 1994; Hillocks, 1986). The reliability of unitization is assessed using Guetzkow's *U* (Guetzkow, 1950).

Tabulation of Term Frequencies

Next, the frequency of occurrence of each unique term in the collection is computed. Within the TM this is achieved using a GAWK script (Robbins, Close, Rubin & Stallman, 1992, p. 164). This frequency information is used for two purposes. First, it is used to decide how many of the most common words should be removed from the collection (Goldsmith et al., 1996). It is also used to calculate weights for each term. Both of these issues are discussed in more depth below.

Removal of Common Words

Not all terms in a collection are equally useful for retrieving or clustering documents. Because grammatical function words (e.g., *a*, *the*, *an*, *and*, *not*, *but*, *is*, *are*, etc.) occur frequently in almost all English documents, the presence of these words carries little useful information. Standard lists of common words (called stop or drop lists) are readily available (Frakes & Baeza-Yates, 1992). Such words are ordinarily deleted before documents are analyzed. In most analyses to date using the TM, custom stop lists have been constructed by removing the _n_ most frequently occurring words from the collection. A somewhat arbitrarily chosen value of n=20 has been used frequently in our experiments with the TM, although Goldsmith has recently devised more principled methods for choosing the number of common words to remove (Goldsmith et al., 1996).

Reduction of Terms to Stem Forms

The total number of unique words in a document can be reduced, and automated analysis of texts can be improved, by removing suffixes from the words that remain after common words have been deleted from a collection of documents (Frakes, 1992). This operation is referred to as stemming because it reduces words to their lexical stems. A single stem normally takes the place of several full terms. The stemming algorithm used here is a variant of the Porter stemming algorithm, which removes common suffixes (e.g., _-s_, _-es_, _-ing_, _-tion_, _-ed_) (Frakes, 1992; Porter, 1980). Both the stemming and the stopping algorithms are available for public use (Frakes, 1995).

Assignment of Term Weights

After common words were removed and the remaining words were reduced to their stem forms, each document is represented as a vector of numerical values. Although it is possible to use a term’s raw frequency as its value in a document vector, more sophisticated term weighting schemes substantially improve the automatic analysis of texts. For the purpose of clustering and discrimination, a term is useful if it occurs frequently in certain documents but rarely in others. Using a LISP computer program written by the author, term weights were calculated according to a standard inverse document frequency formula (IDF) (Harman, 1992; Sparck Jones, 1979):

$$\mathrm{IDF}_i = \log_2 \frac{\max n}{n_i} + 1 \qquad (1)$$

where maxn is the maximum frequency of any term in the collection and n_i is the overall frequency of the ith term in the collection. This function yields high weights for terms that occur rarely and low weights for terms that occur frequently.

Creation of an Inverted File

Before this stage, the collection is represented as a list of documents, and associated with each document is a list of terms (and term weights) that occur in that document. In subsequent computations, however, it is useful to have an inverted index or inverted file. An inverted file is a list of all the unique terms in the collection. Associated with each term is a list of the documents in which that term occurs at least once (Harman, Fox, Baeza-Yates & Lee, 1992). Inverted files are created from weighted document vectors using a LISP program written by the author.

Computing the Interdocument Similarity Matrix

After the documents have been represented as vectors of term weights and an inverted file has been created, a matrix of interdocument similarities is computed. The similarity metric used in clustering should yield high values when two elements had many significant words in common and low values when there were few words in common. The cosine of weighted term vectors, a measure of document similarity taken from the information retrieval literature, had the desired properties (Harman, 1992):

$$\mathrm{Cosine}(x, y) = \frac{\sum_{i=1}^{n} x_i\, y_i}{\sqrt{\left(\sum_{i=1}^{n} x_i^2\right)\left(\sum_{i=1}^{n} y_i^2\right)}} \qquad (2)$$

where x_i and y_i are term weights from documents x and y, and n is the number of unique terms in the collection. This measure computes the cosine of the angle between normalized (unit) vectors of term weights (Harman, 1992). There are, of course, many other similarity metrics that could have been used (Rasmussen, 1992).

For TM experiments, the cosine-based interdocument similarities have been computed in two different ways. Initially, interdocument similarities were computed by the cosine measure using the Proximities procedure in SPSS (SPSS, 1990, p. 553). Subsequently, they were computed using a LISP program written by the author. Each choice has advantages and disadvantages. The advantage of using SPSS is that SPSS is readily available to most users, and no knowledge of LISP programming is required by the user. The disadvantage of using SPSS has to do with computational efficiency. For a collection of N documents, there will be N^2 similarities to compute. Obviously, then, when N grows large, the amount of computation required becomes prohibitive. Fortunately, the similarity matrix is symmetric (i.e., the similarity between document 1 and document 2 is the same as the similarity between document 2 and document 1), and similarities between a document and itself need not be computed. Thus, the number of distinct similarities to compute is actually $(N^2-N)/2$. Still further efficiency is gained because there is no need to compute zero-valued similarities. The inverted index generated previously can be used to guide the computation of interdocument similarities such that only nonzero similarities are computed (Rasmussen, 1992; Voorhees, 1986; Willett, 1980; Willett, 1988). The similarity matrix for most collections will contain relatively few nonzero similarities, so when a specially designed algorithm is used, the time and space efficiency of the similarity computations is much greater than when an off-the-shelf program like SPSS used. The disadvantages of using a special program to compute interdocument similarities are that one must have some knowledge of programming, and one must have access to a computer with the appropriate compiler.

Clustering Documents

Once the proximity matrix had been calculated, the documents were clustered. Several approaches to clustering have been used during the development of the TM. Early on, the output of SPSS Proximities was used as the input to SPSS Cluster. The clustering method was average linkage within groups (also called group average linkage method). This method created clusters that maximized average within-cluster similarity (SPSS, Inc., 1990). SPSS Cluster requires one to specify how many clusters should be created. This was regarded as a decided disadvantage. The output of SPSS cluster was a cluster schedule, which was postprocessed by a LISP program written by the author so that clusters of independent clauses would be output.

Prototype Implementations

During the past year, three different versions of the TM have been implemented, each with progressively more sophisticated clustering capability. The first two prototypes did non-hierarchical or iterative clustering of documents. In the first version, documents were processed sequentially. The first document was assigned to the first cluster. The similarity between the second document and the first was then computed, and if the similarity exceeded a user-specified threshold, the second document was merged into the first cluster. Similarity was measured by the group average method. That is, the similarity between a new document and an existing cluster was defined as the average pairwise similarity of that document with all the documents in the existing cluster. Each of the remaining documents in the collection was processed in the same way. The system stopped merging documents into the first cluster when no more documents had an average pairwise similarity with the existing cluster members that exceeded the threshold. The system then selected the first remaining unclustered document in the collection, defined it as the seed of cluster 2, and began the process again. The process ended when no remaining cluster-document or document-document pairs had a similarity exceeding the threshold.

Although this method demonstrated a crude ability to cluster related documents together, it was flawed in several ways. Perhaps the most significant flaw was that it processed documents in sequential order. A much better strategy would be to search at each step of the algorithm for the nearest document to the current cluster, and then to merge that document. That is the approach taken by the second prototype version of the TM. In this case, the algorithm began by searching the similarity matrix for the closest pair of documents (with ties broken arbitrarily). This pair served as the seed for cluster 1. Then the similarity matrix was searched again for the document whose pairwise similarity to the two documents in cluster 1 was greatest. If the average pairwise similarity for this document exceeded the threshold, it was merged. The process of growing cluster 1 continued until the similarity to the nearest neighbor was smaller than the threshold. Then, the similarity matrix was searched again for closest remaining pair of documents. These two become the seed for cluster 2, the process of adding to cluster 2 proceeds as with cluster 1. The clustering process ceased when no document-cluster or document-document similarities exceeded the threshold.

This best-first iterative partitioning method worked fairly well, and it is the system that most of the current TM analyses are based on. However, it was also flawed because it was not hierarchical. Most standard approaches to document clustering are the so-called hierarchical agglomerative methods

(Willett, 1988). In a hierarchical agglomerative approach to clustering, clusters can merge with individual documents or with other clusters. The versions of the TM just described could only merge a cluster and an individual document, but they could not merge one cluster with another. The two methods described thus far are also affected by the sequence of presentation of documents, which is a highly undesirable characteristic (Willett, 1988).

The most recent implementation of the TM is based on Voorhees hierarchical agglomerative group average linkage method (Voorhees, 1986). This version implements a bona fide hierarchical agglomerative clustering method, and it is not affected by the sequential ordering of the input data. This method is governed by three very simple rules: (a) merge the two closest objects (where an object can be an individual document or a cluster of documents), (b) unless no more objects exist, go to (a), and (c) if no more objects exist, stop. These rules are the same for all hierarchical agglomerative methods. The standard hierarchical agglomerative methods (single linkage, group average linkage, and complete linkage) differ from one another only in the way that they define similarity. Single linkage defines similarity as the <u>maximum</u> similarity between any member of cluster 1 and any member of cluster 2. Group average linkage defines similarity as the <u>average pairwise</u> similarity between all documents in cluster 1 and all documents in cluster 2. Complete linkage defines similarity as the <u>minimum</u> similarity between any member of cluster 1 and any member of cluster 2 (Rasmussen, 1992; Voorhees, 1986; Willett, 1988).

To increase efficiency, the current implementation of the hierarchical agglomerative group average method uses inner product rather than vector cosine similarities, and the quality of the resulting clusters suffers somewhat from the lack of length normalization. An effort is currently underway to implement a prototype that begins with cosine normalized document vectors or to implement the method with cosine as the similarity metric (in spite of the inefficiency).

Integrating the Theme Machine into the Study of Message Design

Most of this essay has been devoted to a theoretical justification for the TM and a summary description of the implemented systems. Before closing, it is important to consider briefly how the TM should be integrated into the systematic study of message design (for a more detailed discussion of these issues, see O'Keefe, 1996). It is important to remember at the outset that the TM is simply a tool for automatic coding. The TM is not a theory and it is not magic. Before one begins to use the TM, one must have a research question worth investigating, and one must have data that could be transcribed into computer readable format. More importantly, one must have a functional (or rational) analysis of the task domain being studied. The TM was explicitly designed to produce structural categories. All of the burden of rational reconstruction and functional analysis falls on the researcher (Craig & Tracy, 1996). The TM simply renders the discourse into a categorical representation that is likely to facilitate subsequent quantitative analysis. Depending on the questions being asked, the researcher must collect data on goals, intentions, or effects in order to have all of the pieces that are needed to study message design. (Recall that message design involves the relationship between message structure and message function. The TM deals only with message structure.) Once the TM has identified a set of structural categories (referred to variously as themes or idea-types), then the original discourse can be represented in terms of these categorical variables, where each variable corresponds to the presence/absence or degree of elaboration of a given theme. Finally, goals or message effects can be analyzed in relation to these categorical variables, and one can get a glimpse of the empirical mapping between message structures and message functions (see, e.g., Lambert, 1996). In the end, of course, the investigator must explain why certain message structures map to certain message functions. This stage of the process is unlikely to be automated in the near future.

Conclusion

In order to formulate and investigate non-circular questions about message design, one must have independent representations of message structure and message function. This essay has described the Theme Machine, a system that does automatic content analysis via document clustering. The structural categories identified by the Theme Machine are based only on the co-occurrence of words in related documents, and are thus completely independent of any functional characterization. Thus, the Theme Machine appears to be a useful tool in the study of message design. The system is still under development, and the existing prototypes suffer from a variety of weaknesses in terms of both efficiency and effectiveness. Nevertheless, preliminary results are good enough to encourage us to pursue the techniques further. Moreover, there is reason to believe that document clustering and related techniques for computer-assisted analysis of text data will prove to be important tools for content analysts in the next century. It is hoped that this essay will encourage others to learn about and experiment with the Theme Machine and related systems.

Appendix

The Theme Machine is not yet available for public use, but the plan is to make it available for non-profit, research use in the near future. To use the Theme Machine, one must have a variety of computational tools, many of which are freely available on the Internet. This appendix lists the most important pieces of software along with brief descriptions and their locations. In addition, much of the preprocessing done for Theme Machine analyses relies on text processing utilities that are most commonly available under the Unix operating system. Thus, it is recommended that Theme Machine analyses be done in a Unix environment.

1. CLISP: CLISP is a public domain (freely available) implementation of Common Lisp the Language. CLISP is available for a variety of platforms. The Theme Machine will run under CLISP. For more information and/or a copy of CLISP, use anonymous ftp from ma2s2.mathematik.uni-karlsruhe.de:/pub/lisp/clisp/.
2. GCL: GNU Common LISP is another freely available LISP system. For more information, use anonymous ftp from ftp.ma.utexas.edu:pub/gcl/.
3. GAWK: GAWK is a powerful and easy to use text processing system for the Unix operating system. Source code for GAWK can be obtained by anonymous ftp from prep.ai.mit.edu:pub/gnu/gawk.
4. Stemming and Stopping algorithms: Source code for the stemming and stopping algorithms used in the Theme Machine, written in the C language, is available by anonymous ftp from ftp.vt.edu:pub/reuse/IR.code/ir-code.

References

Appelt, D. E. (1985). Planning English sentences. Cambridge: Cambridge University Press.

Bonito, J. (1996, May 24). Topical contributions to group discussions: Assessing the contribution of topic knowledge to participation. Paper presented at the International Communication Association, Chicago, IL.

Chafe, W. (1994). Discourse, consciousness, and time: The flow and displacement of conscious experience in speaking and writing. Chicago, IL: University of Chicago.

Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20, 37-46.

Cohen, P. R., & Perrault, C. R. (1979). Elements of a plan-based theory of speech acts. Cognitive Science, 3, 177-212.

Craig, R. T., & Tracy, K. (1996). Grounded practical theory: The case of intellectual discussion. Communication Theory, 5, 248-272.

Frakes, W. B. (1992). Stemming algorithms. In W. B. Frakes & R. Baeza-Yates (Eds.), Information retrieval: Data structures and algorithms, (pp. 131-160). Englewood Cliffs, NJ: Prentice-Hall.

Frakes, W. B. (1995). Information retrieval source code. [On-line]. Available FTP: ftp.vt.edu Directory: /pub/reuse/ir-code Files: stemmer, stopper, stoplist.

Frakes, W. B., & Baeza-Yates, R. (Eds.). (1992). Information retrieval: Data structures and algorithms. Englewood Cliffs, NJ: Prentice-Hall.

Goldsmith, D. J., Dun, S., MacGeorge, E., & McDermott, V. (1996, May 24). Coding topical focus in supportive messages. Paper presented at the International Communication Association, Chicago, IL.

Griffiths, A., Robinson, L. A., & Willett, P. (1984). Hierarchic agglomerative clustering methods for automatic document classification. Journal of Documentation, 40, 175-205.

Guetzkow, H. (1950). Unitizing and categorizing problems in coding qualitative data. Journal of Clinical Psychology, 6(47-58).

Harman, D. (1992). Ranking algorithms. In W. B. Frakes & R. Baeza-Yates (Eds.), Information retrieval: Data structures and algorithms, (pp. 363-392). Englewood Cliffs, NJ: Prentice Hall.

Harman, D., Fox, E., Baeza-Yates, R., & Lee, W. (1992). Inverted files. In W. B. Frakes & R. Baeza-Yates (Eds.), Information retrieval: Data structures and algorithms, (pp. 28-43). Englewood Cliffs, NJ: Prentice-Hall.

Hillocks, G., Jr. (1986). Research on written composition: New directions for teaching. Urbana, IL: ERIC Clearinghouse.

Hovy, E. (1990). Pragmatics and natural language generation. Artificial Intelligence, 43, 153-197.

Jardine, N., & van Rijsbergen, C. J. (1971). The use of hierarchic clustering in information retrieval. Information Storage and Retrieval, 7, 217-240.

Kellerman, K., & Cole, T. (1994). Classifying compliance gaining messages: Taxonomic disorder and strategic confusion. Communication Theory, 4, 3-60.

Lambert, B. L. (1992). A connectionist model of message design. Unpublished Doctoral Dissertation, University of Illinois at Urbana-Champaign.

Lambert, B. L. (1995). Directness and deference in pharmacy students' messages to physicians. Social Science & Medicine, 40(4), 545-555.

Lambert, B. L. (1996). Assessing patient impressions of pharmacists' messages using techniques for semi-automated content analysis. Manuscript submitted for publication.

Lambert, B. L. (in press). Face and politeness in pharmacist-physician interaction. Social Science and Medicine.

Lambert, B. L., & Gillespie, J. L. (1994). Patient perceptions of pharmacy students' hypertension compliance gaining messages: Effects of message design logic and content themes. Health Communication, 6(4), 311-325.

Levelt, W. J. M. (1989). Speaking: From intention to articulation. Cambridge, MA: MIT Press.

O'Keefe, B. J. (1996, May 24). Automated content analysis: Issues of reliability and validity. Paper presented at the International Communication Association, Chicago, IL.

O'Keefe, B. J., & Lambert, B. L. (1995). Managing the flow of ideas: A local management approach to message design. In B. Burleson (Ed.), Communication Yearbook 18, (pp. 54-82). Newbury Park, CA: Sage.

Porter, M. F. (1980). An algorithm for suffix stripping. Program, 14(3), 130-137.

Rasmussen, E. (1992). Clustering algorithms. In W. B. Frakes & R. Baeza-Yates (Eds.), Information retrieval: Data structures and algorithms, (pp. 419-442). Englewood Cliffs, NJ: Prentice-Hall.

Robbins, A. D., Close, D. B., Rubin, P. H., & Stallman, R. M. (1992). The GAWK manual. Cambridge, MA: The Free Software Foundation.

Saeki, M., & O’Keefe, B. J. (1994). Refusals and rejections: Designing messages to serve multiple goals. Human Communication Research, 21, 67-102.

Salton, G., Allan, J., Buckley, C., & Singhal, A. (1994, June 3). Automatic analysis, theme generation, and summarization of machine-readable texts. Science, 264, 1421-1426.

Salton, G., & McGill, M. (1983). Introduction to modern information retrieval. New York: McGraw-Hill.

Sparck Jones, K. (1979). Search term relevance weighting given little relevance information. Journal of Documentation, 35, 30-48.

SPSS, I. (1990). SPSS® Reference Guide. Chicago, IL: SPSS, Inc.

Voorhees, E. (1986). Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. Information Processing and Management, 22(6), 465-476.

Willett, P. (1980). Document clustering using an inverted file approach. Journal of Information Science, 2, 223-231.

Willett, P. (1983). Similarity coefficients and weighting functions for automatic document classification: An empirical comparison. International Classification, 10, 138-142.

Willett, P. (1988). Recent trends in hierarchic document retrieval: A critical review. Information Processing and Management, 24, 577-597.