DOCUMENT RESUME

ED 144 590 IR 005 212

AUTHOR Becker, David S.; Pyrce, Sharon R.
TITLE Enhancing the Retrieval Effectiveness of Large Information Systems. Final Report for the Period 1 June 1975-31 December 1976.
INSTITUTION Illinois Inst. of Tech., Chicago. Research Inst.
SPONS AGENCY National Science Foundation, Washington, D.C. Div. of Science Information.
REPORT NO IITRI C6345
PUB DATE 31 Jan 77
GRANT SIS-75-16262
NOTE 170p.; Best copy available

EDRS PRICE MF-$0.83 HC-$8.69 Plus Postage.
DESCRIPTORS Algorithms; *Bibliographic Coupling; Cluster Grouping; *Computers; Electronic Data Processing; Experiments; Graphs; Information Processing; Information Retrieval; *Machine Translation; *On Line Systems; Programing Languages; *Relevance (Information Retrieval); Tables (Data)
IDENTIFIERS Automatic Term Classification; Boolean Search Strategy; CA Condensates; COMPENDEX

ABSTRACT

The goal of this project was to find ways of enhancing the efficiency of searching machine readable data bases. Ways are sought to transfer to the computer some of the tasks that are normally performed by the user, i.e., to further automate information retrieval. Four experiments were conducted to test the feasibility of a sequential processing hypothesis: a multi-step search process using Boolean search as the first step and subject term clustering as the second. The multi-step processing can be further strengthened by incorporating some semantic information into statistical string processing by the use of a new method of Automatic Term Classification (ATC). The results suggest an organization for information retrieval systems of the future in which several processing techniques are used during a single retrieval. Charts, tables, figures, and statistical data for the experiments are included. Appendices include all symbols used during the experiment; probability of term match formulas, computer programs used in the experiments; and sample mappings of selected words. The data bases used were selected files of Chemical Abstracts Services CACon and Engineering Index COMPENDEX. (Author/JPF)

NSF GRANT NO. SIS 75-16262
PROJECT NO. IITRI C6345

ENHANCING THE RETRIEVAL EFFECTIVENESS OF LARGE INFORMATION SYSTEMS

FINAL REPORT FOR THE PERIOD 1 JUNE 1975 - 31 DECEMBER 1976

PREPARED FOR:
NATIONAL SCIENCE FOUNDATION
DIVISION OF SCIENCE INFORMATION
1800 G STREET, N.W.
WASHINGTON, DC 20550

PREPARED BY:
DAVID S. BECKER AND
SHARON R. PYRCE
IIT RESEARCH INSTITUTE
10 WEST 35TH STREET
CHICAGO, IL 60616

31 JANUARY 1977

ACKNOWLEDGEMENT

The authors want to thank several people who contributed to the program. Scott E. Preece wrote much of the clustering software and participated in the design of the program. A number of key suggestions were made by Martha E. Williams. The work draws upon the software and the user files that were created under the operation of the Computer Search Center (CSC) of the IIT Research Institute. The CSC was the result of several years of development by a team of many individuals. The current research and the CSC activities were performed under the sponsorship of the National Science Foundation, Division of Science Information.

ABSTRACT

The goal of this project is to find ways of enhancing the efficiency of searching large machine-readable data bases. This includes improving the recall and precision characteristics of retrievals initiated by user requests as well as helping the user to form concepts. For the latter, ways are to be sought to transfer to the computer some of the tasks that are normally performed by the user, i.e. to further automate IR (information retrieval). Such developments are motivated by the rapid growth in the volume of on-line IR activities, and the fact that the cost of searches is no longer limited by cpu search costs. Rather, it is limited by labor costs (profiling, evaluating output, bookkeeping, etc.) and I/O costs (printing, mailing, etc.). For a typical search, costing between 100 and 300 dollars, usually less than $5.00 in cpu is consumed. Such costs suggest that large efficiency gains can be made by further automating IR systems functions. Underlying these goals are two general issues. The first is the relationship between statistical string processing and semantic word processing. The second is the concept of multi-step processing of a search request.

Statistical string processing pertains to those IR functions that can be performed without knowing the definitions of the terms (character strings), i.e. sorting terms and grouping records on the basis of the terms they contain. This is the typical method used in Boolean searches and simple term clustering.

Semantic word processing pertains to those word relationships that depend on term definitions, i.e. the meaning of the term in the context of the data base. Multi-step processing of large files involves using more than one methodology in distinct steps, to process a single search request. The steps are arranged so that the first process is most appropriate for

use on a very large file. The second step then operates on a subfile identified by the first step and further refines the output file, etc. In this study, the multi-step search idea was tested at length, using Boolean search as the first step and subject term clustering as the second. The results were encouraging. Moreover, it was found that the processing may be further strengthened by incorporating some semantic information into statistical string processing by the use of a new method of Automatic Term Classification (ATC). The ATC method allows the string comparison mechanism to either match the categories rather than match the strings, or to limit the compares to those terms that lie within a given category. The latter process is new, and corresponds to the psychological process of focusing attention on a limited family of record aspects. Overall, the results suggest an organization for the IR system of the future in which several processing techniques are used during a single retrieval, and in which the system will be an active search partner performing like an ideal librarian.

TABLE OF CONTENTS

FIGURES

FIGURES (Continued)

TABLES

"The mind requires a representation of knowledge wherein interassociated ideas are labeled according to their type. Such labeling seems utterly necessary in order to direct efficient searches through memory for information that meets certain requirements..."[1]

ENHANCING THE RETRIEVAL EFFECTIVENESS OF LARGE INFORMATION SYSTEMS

1. BACKGROUND

During the past 20 years, the application of computer technology to solving information retrieval (IR) problems has become commonplace. These applications are motivated by many factors, the most prominent of which are probably the advances in electronic data processing and computer print setting technology, the information explosion and the recognition by agencies, primarily the National Science Foundation (NSF), that the cost-benefit ratios favoring research on IR technology are enormous.

To date, the commercially viable IR systems for large bibliographic data bases have not been "thinking" systems, in the sense that they identify records for a retrieval based on the character strings that they contain - independent of the conceptual definitions of those strings. For instance, one may query Boolean systems for co-occurrences (occurrences within one record) of the strings "ozone" and "tomato" in order to identify those records pertinent to the concept "the effect of ozone on the tomato plant." In performing this search, the system does not make use of the definition of ozone as a molecule composed of three oxygen atoms nor does it use the definition of tomato. Rather, the system merely searches for occurrences of the explicit character strings, "ozone" and "tomato". The systems of most organizations work this way, including the IIT Research Institute's Computer Search Center, (CSC), The National Library of Medicine (NLM), Lockheed Information Systems, SDC Search Service and the University of Georgia Information Dissemination Center. One exception is the Institute of Scientific Information (ISI) system, which identifies related records via references cited within each document.[2] That is, the ISI system effectively sidesteps the problems of handling and manipulating subject terms by linking each record to those records that it cites. In the future, it would be desirable to co-ordinate this capability, which is a natural extension of manual procedures, with the subject term oriented capabilities studied in this report.

The enormous success of IR systems based on merely matching character strings motivates one to try to automate more of the steps in the IR process, conceptually outlined on Table 1. The task of composing a combination of character strings that will represent a given concept (profiling) and retrieve appropriate records with good performance is difficult. It requires knowledge of the statistics of the terms within the data base as well as knowledge about the desired concept. Accordingly, the profiling task is usually performed by information specialists. Search failures can occur for many reasons, including: failure to translate the concept into the specific terminology of the system, failure to identify closely related concepts and failure to learn during the course of the search those new concepts that are related to the old one by implications - rather than overlap of character strings.

Clearly, some of the capabilities that one would like to automate in an IR system are those of an ideal librarian: the ability to summarize the general characteristics of a retrieval or a collection without necessarily having to analyze all the implications of the text in the records, the ability to disambiguate different classes of term co-occurrences (i.e. distinguish between "the effect of ozone on tomato plants" and "the generation of ozone by tomato plants"), the ability to suggest to the user certain aspects of the search that are likely to be of interest, etc. Because these capabilities involve using terms as more than just character strings, they imply that the system will have to have available to it, some

STEPS	MANUAL SEARCH OF CACon	CSC SEARCH OF CACon
The User		
1. Conceptualizes document characteristics	Identifies known authors, corporate authors, subject areas, related concepts, time periods...	Same
2. Expresses characteristics in terms of Data Base and IR system	Identifies key words and subject index terms with the subject areas, identifies relevant CA section numbers. Adjusts time period for publication lag...	Same plus association of keywords and keyword fragments in logic statements, examination of keyword and fragment frequencies...
3. Operates system and receives output	Refers to CAS Subject Index, Formula Index, Subject Guide and Author Index for abstract numbers. Proceed to abstracts for references...	Key input and operate computer system. Output computer printed citation cards, sometimes obtain full abstracts for references
4. Evaluates output and	Reads parts or all of abstracts and makes decisions as to completeness and relevance/.	Same
4a. Is satisfied, or	Decides that search has exhausted CAS capabilities and/or has fulfilled search needs.	Same
4b. Modifies expression, or	Includes related terms, corrects errors of translation..returns to Step 3.	Same
4c. Modifies concept, or	Corrects errors of thought or incorporates new ideas learned from search. Returns to step 2.	Same
4d. Terminates unsatisfied	Is frustrated, runs out of time or money....	Same

TABLE 1. Steps in Information Retrieval

degree of conceptual term definition. Language processing using conceptual term representation is usually called semantic information processing.

Curiously, it has been found that attempts to incorporate semantic information into an information retrieval search mechanism have generally resulted in degradation of search retrieval performance for equal search cost, as compared with statistical string processing.[3,4] That is, for a given dollar cost, a statistical string based search mechanism will generally give better performance than a system using semantic information.

Many of the attempts to incorporate a degree of semantic information into IR systems have been reviewed by Montgomery[5], and more recently by Damerau[6]. The general structure of these systems is shown in Figure 1, adapted from Montgomery[5].

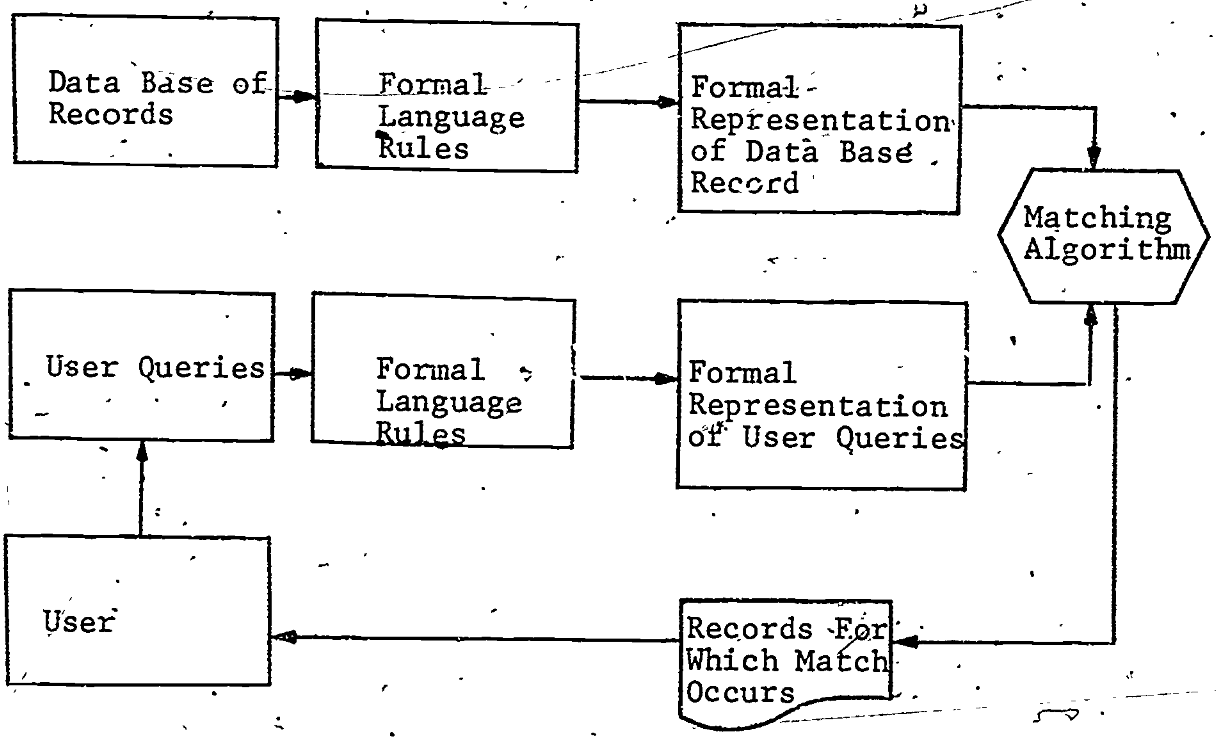


Figure 1. IR Systems Design Based on Canonical Representation

User queries and data base records are each translated into a formal representation that facilitates the recognition of matches between them. The choices for the format representation vary widely, including contributions from semantics and syntax of the data contents. Some systems, such as those of Sager[7] and Kuno-Oettinger[8] use a syntax-driven phrase structure grammar to identify and rewrite records into canonical forms. These systems are top-down in the sense that they used fixed rules to classify input strings. Transformational grammars have also been applied.[9] Other systems use semantics-driven procedures to replace records with a representation in semantic primitives. The systems of Wilks[10] and Laffal[11] are of this type. Yet other systems combine syntactic and semantic information to approach a more complete representation of the data base. Von Glaserfield's[12] system is of this type. Finally, there are more comprehensive Artificial Intelligence (AI) systems, like those of Simmons[13], Schank[14] and Winograd[15], which use internal representations that approach the power of handling text in a cognitively meaningful manner. Such systems, of course, are much more expensive to operate because of their high requirements for computer memory space and processing time. However, their capabilities are impressive. AI Systems exist today that can input up to about one short paragraph of English text, in a very limited context of discourse, can process it into an internal representation and then can answer questions about it, phrased in nearly free English. The existence of such systems today motivates the question of what their relationship will be to the IR system of the future. That is, are the statistical string techniques that are dominant today at commercial search services destined to be replaced by semantic techniques in the future, or is a sharing of roles more likely?

Because statistical processes have been most cost-efficient, research has recently been done on enhancing the efficiency of these processes. A logical extension of the Boolean search procedure is to relate the probability of conceptual similarity between two records to the number of character strings that they hold in common. That is, records containing the same strings are more likely to concern the same concepts than are records that don't. Using this principle, it is possible to partition record collections into groups, or clusters, such that members within a group share vocabulary overlap, and, probably, concepts. Unfortunately, the cost of clustering increases rapidly as the file size increases, because it involves comparisons among all records. For a collection of N_F* records, most clustering algorithms consume an amount of computer processing time proportional to between $N_F \cdot \ln(N_F)$ and N_F^2.

For instance, if a file with 100 records is clustered using 10 cpu, then a file of 10,000 records would require between 400 cpu and 10^5 cpu. Since many bibliographic files are much larger than 10,000 records, it is difficult to see how a clustering algorithm could be efficiently used on a large file during a single on-line accession.

Using clustering on small sets, many investigators, principally G. Salton[16] and K. Sparck-Jones[17], have studied new designs for IR systems. Salton generally uses about 1,000 records, and K. Sparck-Jones uses fewer. Via this method, records are clustered into groups before retrievals are done. Then, a user query may retrieve any of the already clustered record groups. This process is analagous to retrieving all entries under a subject category such as a Library of Congress Catalog number. However, with clustering, the records may be conveniently ranked according to their probable relevance to the search query. One feature of these systems that has recently been exploited is that user judgements on relevancy of output may be readily incorporated, by automatic means, back into the retrieval mechanism so as to re-prioritize the output.[18,19] That is, if a given record is rated as relevant, the terms in that record can be more highly associated with relevance and the terms not appearing can be more highly associated with non-relevance. The opposite procedure is applied for records judged to be non-relevant. The results of these judgements are then applied to all candidate records, through the terms they contain. Such procedures are capable of very high IR performance in situations where many relevance judgements may be accumulated. In contrast,

*All symbols used are defined in Appendix A.

it seems that for the case of on-line interactive retrieval, it would be more efficient to have the searcher make the judgements directly on the terms themselves. Then, the system does not need a procedure to automatically weight the terms. Instead, it is told that information directly. The key points developed by these workers that are relevant to the work to be discussed herein are:

1. Statistical methods exist for automatic partitioning of records into classes based on their term overlap;
2. Clustering can either be user independent or user dependent; and
3. Subject term clustering is usually limited in application to small files for reasons of processing cost.
4. User relevance judgements made on one group of records can be automatically extrapolated to another group of records on the basis of their shared terms.

PROGRAM CONCEPT

The central idea of this program is that more than one search methodology can be used during the course of a single retrieval. Perhaps it is the case that IR systems incorporating some degree of semantic information processing are less successful than purely statistical string processing programs because the statistical processing is the most efficient single way to conduct a retrieval. That is, perhaps the various retrieval methodologies can be thought of as screens of varying coarseness, with Boolean string matching being nearly the most crude, clustering, for example, being less crude (because it uses all of a record's terms, rather than only the selected ones as occurs for Boolean search), and semantic information processing being much finer. If the screen analogy is valid, then the most cost-effective way to perform a very precise search is not to apply the finest screen to every record. Rather, it is to start with a coarse screen, and to use it to separate out all those items that, at its level of coarseness, do not apply, and then to apply the more fine screens to the remaining items. This implies that the many forms of canonical representation previous alluded to, and their corresponding match mechanisms, are all candidates for use in co-operative systems more complex than that shown in Figure 1. That is, any combination of those systems could be arranged in a sequence of steps to process a single user query. Many combinations are attractive. For this study, Boolean searching was chosen as the first step of an information retrieval, and subject term clustering of the resultant set of (Boolean search selected) records was chosen as the second step.

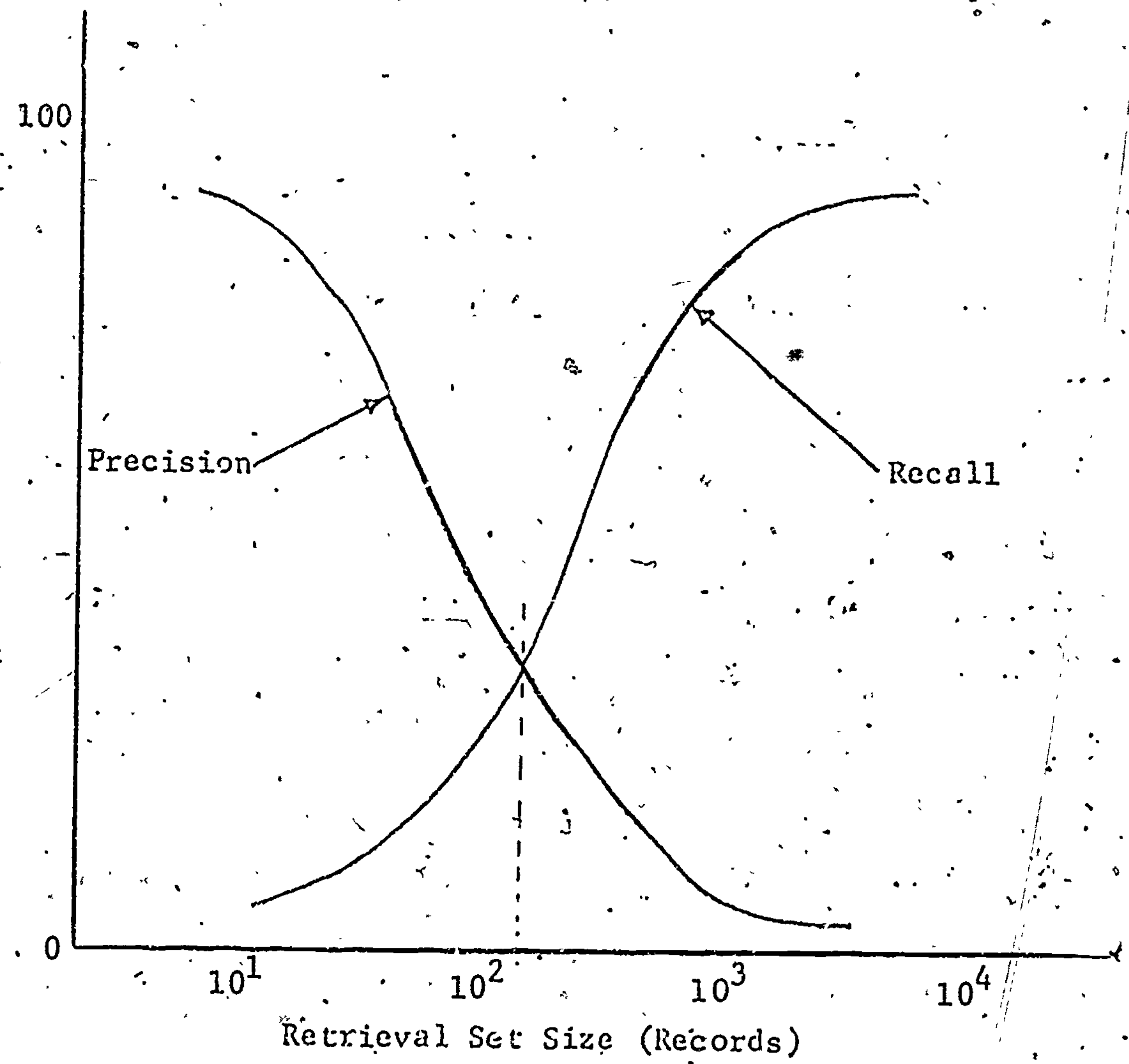
There are several factors that motivate the coupling of a Boolean first step with a clustering second step. First, Boolean techniques work well with inverted term files, so that they easily accomodate large files. Subject term clustering techniques, however, are prohibitively expensive for large files. Second, whereas Boolean techniques require user specified terms, cluster techniques work on the contents of records, and so can accomodate the many highly specific low frequency terms that are so inaccessible to Boolean methods in producing the pattern. Also, because clustering operates on the record contents, and, in effect, summarizes the retrieval as a pattern, the pattern can assist user concept formation about the term co-ordinations that are represented in the retrieval. That is, IR is essentially a closed problem because the user can always sidestep the IR system and manually screen all the records for the desired properties. Hence, the measure of the effectiveness of any IR system is the degree to which it reduces the number of user judgements while preserving sufficient recall. By grouping Boolean-retrieved records, clustering can reduce the number of user decisions required to the number of clustered groups. That is, if all records in a group are similar, then only one or two of them need to be examined so as to evaluate the relevance of all the members of the group. Second, the grouping provides a mechanism for feeding back to the user summary level information about the characteristics of his retrieval set. For such a mechanism to be useful it should perform at a cost less than that which would be required for manual evaluation of the retrieved set or other available means.

Some might argue that it would be more appropriate to couple a Boolean first step with a syntax based second step. It was decided to use clustering because content information, which is accessible to clustering methods, seems to be a more coarse screen than syntax information. After all, titles are an effective retrieval field, and titles are usually phrases, not sentences. It seems natural to first consider the terms that are present, then their context, and then their syntax.

THE RETRIEVAL PERFORMANCE PROBLEM - TYPICAL PARAMETERS

The retrieval performance problem involves the difficulty one has in achieving high recall with high precision in, for instance, on-line bibliographic retrievals. This problem is illustrated in Figure 2 for typical search parameters for an on-line retrieval from a large data base. If terms of very high specificity are used in the Boolean retrieval search strategy (i.e. low frequency terms such as the names of specific plants (pine, carrot, etc.)), the number of records that satisfy the search strategy (the retrieval set) is small, the precision is high (most retrieved records are relevant) but many relevant records are not retrieved, because they did not contain the specific terms chosen by the searcher. If, alternatively, terms of low specificity are used in the search strategy (i.e. high frequency terms such as plants, botany, etc.), the number of records that satisfy the search strategy is large, the precision is low (many retrieved records are not relevant), but most relevant records are retrieved. Thus, there is a tradeoff between the number of relevant records missed and the user time required to evaluate possible non-relevant records. For different users, the tradeoff is usually satisfied by varying the size of the retrieval set. In Figure 2, a retrieval of about 100 records results in a precision of about 30%, so that 30 relevant and 70 non-relevant records are retrieved. A more complete search, yielding a retrieval of 1,000 records results in a precision of about 10%, so that about 100 relevant records and 900 non-relevant records are retrieved.

Not all searches need be exhaustive, so not all users will opt for the larger, more complete searches. At IITRI's CSC, however, exhaustive searches are often required, and so the following question arose. Suppose that the Boolean search parameters were arranged to yield an exhaustive retrieval? Is there any additional computer processing that could be performed on the retrieval set so as to further separate the relevant from the non-relevant records? That is, the Boolean search technique, even when used with general terms so as to yield high recall, is still



$$\text{Recall} = \frac{\text{Relevant Records Retrieved}}{\text{Relevant Records in the Data Base}}$$

$$\text{Precision} = \frac{\text{Relevant Records Retrieved}}{\text{Total Records Retrieved}}$$

Figure 2: Typical Recall - Precision Tradeoff as a Function of Retrieval Set Size for Boolean Search Strategies.

a very effective filter, reducing the set of candidate records for retrieval from perhaps 2,000,000 to perhaps 2,000, as illustrated in Figure 3. Now, the 2,000 item retrieval could be further refined by additional Boolean restrictions. The problem is that the formulation of those additional restrictions would be very time-consuming because they would necessarily involve low frequency terms, and hence, a long and complicated search strategy. Also, in order to formulate this long and refined search strategy, it is necessary to find out some of the summary level characteristics of the retrieved set, and the only way to do that now is to scan some of those records or try to guess the terms that are present and to enter them as search terms. However, why should a user have to guess? Wouldn't it be better for the computer to sort the characteristics of the relatively small retrieval set and report them back to the user? The manual scanning process of refining the Boolean logic is so slow that a user is often better off, when he requires an exhaustive search, to simply print the entire high recall set and manually reject the non-relevant items. If the retrieval set of 2,000 records were partitioned into 20 clusters (of 100 records each), and if all of the relevant records were to be in one cluster, then identification of that cluster would yield a high recall search with high precision. The Boolean step would be recall-oriented and the clustering step would be precision-oriented. The selection of the appropriate (high recall with high precision) cluster could then be accomplished by, perhaps, examining one or two sample records from each cluster, reducing the number of relevancy decisions from 2,000 to about 20.

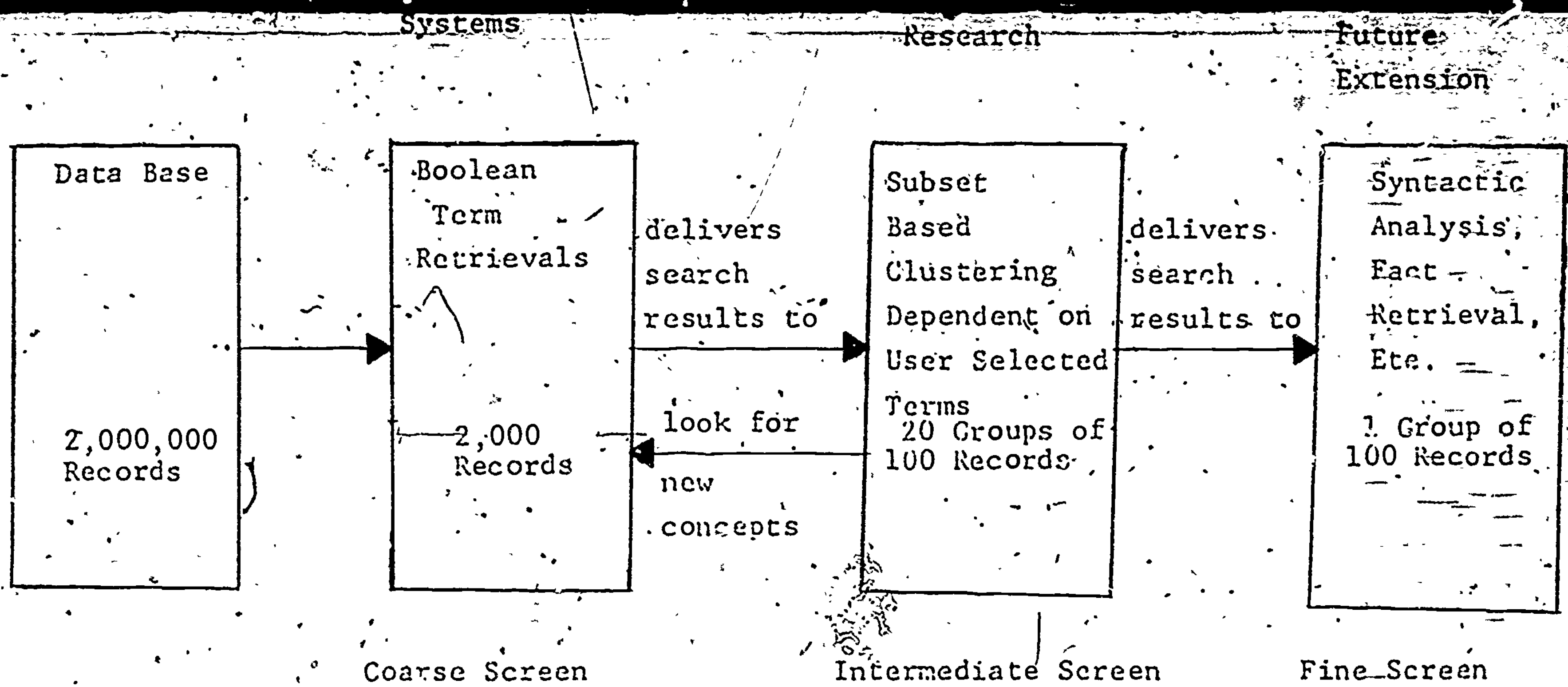


Figure 3. A Multistep IR Processing Stream

2. METHODS AND MATERIALS

DATA BASES

The data bases used for the experiments were Chemical Abstracts Services CACon, Volumes 82 and 83 and Engineering Index's COMPENDEX (Ei), Volumes 74 and 75. CACon addresses wide range of chemistry related literature. It covers about 300,000 references per year, and during this time period, groups them into 5 supersections composed of 80 sections, as illustrated in Figure 4. Each of the 80 sections is further subdivided into subsections. There is a total of about 700 subsections. Individual records are assigned to categories by best fit. Cross-indexing terms indicate when other assignments were considered acceptable. COMPENDEX has a similar structure in that each recrod is assigned to categories (Card-Alert-Codes). However, the codes are applied more in the spirit of controlled indexing, and multiple code assignments to a given record is the rule, rather than the exception. This is opposed to the fact that a given record in CACon is usually assigned to only one section and usually has no cross-indexing terms.

Each record in CACon contains the following fields: CODEN, title, indexing (including section and subsection assignment), bibliographic reference and author. The clustering experiments used the first three of these fields, in various combinations. The COMPENDEX records contained the same fields as CACon, and in addition, also contained full text abstracts. Clustering experiments for COMPENDEX used the abstract field.

ABSTRACT SECTIONS

Biochemistry Sections

Organic Chemistry Sections

ABSTRACT SECTIONS

Macromolecular Chemistry Sections (POST)

Applied Chemistry and Chemical Engineering Sections

Physical and Analytical Chemistry Sections

Figure 4A. CACon Data Base Structure - Sections

Subsection Arrangement for CA23 - Aliphatic Compounds

0. Review
1. General
2. Hydrocarbons
3. Halides
4. Amines, amine oxides, imines, quaternary ammonium compounds
5. Hydroxyl amines, hydrazines, azines, triazines, azides, azo and diazo compounds
6. Nitro and Nitroso Compounds
7. Alcohols and thio alcohols
8. Alcohol esters with inorganic acids including cyanates and isocyanates
9. Ethers and thio ethers
10. Peroxides and hydroperoxides
11. Sulfoxides, sulfones and sulfonium compounds
12. Sulfenic, sulfinic and sulfonic acids
13. Selenium and tellurium
14. Aldehydes and derivatives
15. Ketones and derivatives
16. Carbonylic acids and peroxycarbonylic acids and their sulfur-containing analogs and salts
17. Esters, lactones, anhydrides, acyl peroxides, acyl halides
18. Amides, lactams, amidines, imidic esters, (hydr)azides
19. Nitriles, isonitriles and acylcyanides
20. Ureas, carbonic acids, guanidines, and sulfur containing analogs

Figure 4B. CACon Data Base Structure - Subsections

ORGANIZATION INDEX

400

Civil - Environmental - Geological - Bioengineering

Planning, design, construction and maintenance of fixed structures and facilities; including public works, for community development, environmental control, housing, industrial activity, and transportation.

Group No.	Division No.	$ Annual Subscription

400 — CIVIL ENGINEERING, GENERAL

401 — Bridges and Tunnels $65

Design, construction, maintenance and repair of arch, bascule, cable-stayed, cantilever, composite, lift, movable, plate girder, pontoon, suspension, swing, trestle, truss and other types of bridges of concrete, masonry, steel and other materials for causeway, highway, military, pedestrian, pipeline, railroad and viaduct applications, bridge anchorages, decks, piers, superstructures, and supports, construction of pedestrian, railroad, utility, vehicular, water supply and other tunnels.

402 — Buildings and Towers $100

Design, construction, service equipment, maintenance and repair of apartment, auditorium, commercial, educational, exhibition, factory, farm, garage, industrial, laboratory, medical, office, public, recreational, religious, residential, stadium, store, terminal, theater, warehouse and other buildings; conventional, inflatable, modular, multistory, portable, prefabricated, temporary and other types of building construction, exposition structures, masts, monuments, pylons, silos, stacks, towers and other special structures.

403 — Urban and Regional Planning and Development $65

Design and development of urban areas and regions, including cities, suburbs and towns; land use planning; municipal engineering and public works including provision of facilities and structures for education, government, health, housing, recreation, shopping, and urban transport including internal transport facilities, urban rehabilitation and renewal

404 — Civil Defense and Military Engineering $65

Civilian protective works and shelters, military bases, buildings, construction, equipment and materiel, military research on ballistics, missiles and other ordnance military science, missile sites and systems; naval buildings and structures

405 — Construction Equipment and Methods; Surveying $100

Design and manufacture of blasting equipment, caissons, cofferdams, concrete mixers, construction vehicles, cranes, derricks, dredges, earthmoving equipment, hoisting equipment, piles and pile drivers, pneumatic tools, power shovels and other equipment items, construction operations such as dredging, erection, excavation, grading, grouting, masonry, prefabricated construction, riveting, rock drilling, and shaft sinking techniques of concrete, steel, and timber construction, techniques of surveying and mapping, including photogrammetric methods

406 — Highway Engineering $65

Highways roads and streets engineering including culverts, drainage, embankments, interchanges intersections, lighting, markings, median dividers and guard rails, overpasses and underpasses, railroad-crossings, road stabilization and structural design, roadside improvement, route planning and siting, toll roads and related structures; maintenance of highways and other routes.

407 — Maritime and Port Structures; Rivers and Other Waterways $65

Design, construction, equipment, maintenance and repair of breakwaters, docks, groins, jetties, marine terminals, piers, pontoons, quay walls, revetments, seawalls, shore and harbor protection and coastal engineering structures generally, harbor and port facilities, lake, river and other waterway improvement and regulation by means of dredging, navigation canals, channels, gates and locks; sedimentation and silt control, and bank stabilization.

408 — Structural Design $100

Design, construction and testing of arches, beams, columns, cylinders, disks, domes, framed structures, girders, plates, sheet materials, shells, spheres, struts, trusses and other structural members, sections and shapes; structural stress analysis, photoelasticity and other methods of stress determination in structural design, wind stresses.

410 — CONSTRUCTION MATERIALS

411 — Bituminous Materials $65

Manufacture, testing and use of asphalt, pitch, tar and derivative byproducts for applications such as coatings, flooring, pavements, roads and streets, roofing, sealants and waterproofing.

412 — Concrete $100

Admixtures, aggregates, cement, crushed stone gravel, lime, mortar, ready mix, reinforcing materials, sand and combinations thereof to form concrete products, lightweight concrete, reinforced structures and surfaces including blocks, precast and prestressed units and other structural forms.

413 — Insulating Materials $100

Asbestos, cork, fiber and fiberboard, foam materials, glass, magnesia, mica, mineral wool, plaster and plasterboard, plastics, rubber, vermiculite, wax and other insulating materials as used for acoustical, electrical, flame, moisture, radiation, reflective, sound, thermal, and vibration insulation.

414 — Masonry Materials $65

Basalt, brick, clay, glass, granite, limestone, marble, sandstone, slate, terra cotta, tile and other structural ceramic and stone materials for buildings, engineering works, and structures, mortars.

415 — Metals, Plastics, Wood and Other Structural Materials $65

Aluminum, copper, iron, magnesium, plastics, steel, wood, and other structural materials to form clad, composite, honeycomb, laminated, reinforced or sandwich materials for building and structural use.

420 — MATERIALS PROPERTIES AND TESTING

421 — Strength of Materials; Mechanical Properties $100

Elasticity, plasticity, rheology, stress-strain relations and associated phenomena and properties such as abrasion resistance, crack formation, creep, deformation, ductility, failure, fatigue, fracture, hardness, malleability, radiation damage, strain hardening, strength, surface roughness, wear, yield strength and other mechanical properties, testing of metals in bulk form or as crystals, films, foils, sheets, whiskers, wire and powder metal products; testing of nonmetallics in bulk or divided form or as combinations of materials such as composite, honeycomb, laminated, reinforced and sandwich materials.

422 — Strength of Materials; Test Equipment and Methods $100

Apparatus such as hydraulic impact (e.g. Charpy, Izod), indentation (e.g. Brinell, Rockwell, Vickers), screw-gear and universal machines, and instruments such as extensometers, strain gages and other devices; bending, compression, creep, fatigue, hardness, high and low pressure and temperature, impact, shear, tension, and torsion test methods, nondestructive techniques such as brittle coating, liquid penetrant, magnetic particle, radiographic, ultrasonic, X-ray and similar means for detection of defects and flaws; special techniques for accelerated testing.

423 — Miscellaneous Properties and Tests of Materials $100

Other physical and general properties of materials as determined by miscellaneous test equipment including chemical, electrical, environmental, nuclear, optical, physical and thermal apparatus and instrumentation.

430 — TRANSPORTATION

431 — Air Transportation $65

Air cargo, freight, mail and passenger services, civil and military; aircraft maintenance and repair facilities and methods; airlines, reservation systems, routes, scheduling, airports, buildings, hangars and terminals, ground facilities, markings, runways, air safety, air traffic control, navigation aids.

432 — Highway Transportation $65

Commercial, freight, passenger, public service and other forms of motor transportation employing automobiles, buses, taxis, trailers, and trucks and including operation of fleets, lines, routes and terminals; filling stations, garages, repair shops and vehicle maintenance and repair, highway safety, traffic control, signals and surveys.

433 — Railroad Transportation $65

Freight and passenger rail services and industrial railroads including use of rail-highway containers and trailers, and operation of lines, reservation systems, routes, switchyards and terminals; repair shops and maintenance and repair of rolling stock; safety, signal systems and traffic control.

Figure 5: COMPENDEX Data Base Structure

Group No.	Division No.	$ Annual Subscription

434 — Waterway Transportation $65

Cargo shipment and passenger transportation on coastal, inland, transoceanic or other routes; cargo transfer and terminal operations; marine safety and navigational aids including beacons, buoys, lighthouses, lightships, operation of barges, containerships, ferries, freighters, merchant ships, passenger vessels, tankers, tugs and other craft.

440 — WATER AND WATERWORKS ENGINEERING

441 — Dams and Reservoirs; Hydro Development $65

Design, construction and repair of arch, buttress, earth, embankment, gravity, movable, and rock fill dams, multipurpose and special purpose reservoirs, hydraulic structures associated with dams, and hydro-power development such as channels and chutes, conduits, draft tubes, fishways, flumes, forebays, penstocks, river basin development, siphons, sluice gates, spillways, stilling basins, surge tanks, and weirs.

442 — Flood Control; Land Reclamation $65

Drainage, runoff and subsurface water quantity control; flood routing, flood control measures and structures such as dikes, drainage basins, levees, river embankment works and storage systems, flood forecasting, measures, structures and works for irrigation and reclamation of land

443 — Meteorology $100

Aerology, aeronomy, atmosphere, climatology, cloud formation and seeding, ice, rain, snow, and storm phenomena, weather modification, winds, weather forecasting and measurement by anemometric, barometric, hygrometric, pressure, temperature and other instrumentation including use of meteorological balloons, radiosondes, rain and snow gages, satellites and telemetry systems

444 — Water Resources $65

Surface and underground water occurrence, resources and supplies including aquifers, artesian water, groundwater, springs, water bearing formations and strata, waterfalls, watersheds, water wells, and hydrogeology, water conservation, water law, water prospecting, water yield improvement, regional water resources, hydrological cycle generally including evaporation, precipitation and transpiration of moisture and its influence on atmospheric water vapor, soil moisture, surface water and water table, regional hydrology

445 — Water Treatment, General and Industrial $65

Improvement of water quality for general, potable or process use; methods and equipment designed for aeration, chlorination, coagulation, demineralization, filtration, flocculation, fluorination, sedimentation, softening and other treatment techniques, water analysis, bacteriology, and chemistry; saline water conversion

446 — Waterworks $65

Design, construction, equipment, operation, maintenance and repair of water supply systems including aqueducts, distribution lines, mains and water pipelines generally, municipal water supply and regional waterworks; pumping plants and stations; water tanks, towers and related hydraulic structures; water utility management.

450 — POLLUTION, SANITARY ENGINEERING, WASTES

451 — Air Pollution $100

Engineering and economic aspects of air pollution control; abatement and control of gaseous and particulate pollutants such as dust, engine exhausts, flue gases, fly ash, fumes, odors, smoke and soot; methods and equipment used for air and dust analysis, density measurement and sampling; dust collectors, filters, precipitators and recovery systems; dust hazards and protective devices.

452 — Sewage and Industrial Wastes Treatment $100

Environmental sanitation practices, particularly the disposal, removal and treatment of agricultural, community and industrial sewage, design and development of incinerators for conversion and disposal of solid wastes, recovery of thermal energy, recycling and production of useful byproducts; design, construction, operation, maintenance and repair of sewage treatment plants including equipment such as filters, pumping plants, pumps and tanks; sewers and street sanitation.

453 — Water Pollution $65

Abatement and control of biological, chemical, physical, and thermal pollution of shores, streams and waters generally by industrial process effluents, mine drainage, natural eutrophication, oil spills, radioactive materials, refuse, salt water intrusion, sewage, wastes and other pollutants.

460 — BIOENGINEERING

461 — Biotechnology $100

Engineering aspects of human factor requirements in the design, development and operation of man-machine systems; biomechanics, biomedical measurements, biometrics, bionics, cybernetics, ergonomics, and life-support systems generally.

462 — Medical Engineering and Equipment $100

Devices and instruments for medical practice and research including equipment for specialties such as anesthesiology, cardiology, encephalography, fluoroscopy, instrument patient monitoring, radiology, and surgery; design and manufacture of hospital equipment and facilities; design, manufacture and materials for use in medical supplies such as artificial organs, cardiac pacemakers and valves, dental materials, eyeglasses, hearing aids, prosthetic devices, respirators and therapeutic aids

470 — OCEAN AND UNDERWATER TECHNOLOGY

471 — Marine Science and Oceanography $100

Chemical and physical properties of seawater, currents, ice formation, tides, waves and weather effects, and engineering implications; island formation and erosion; ocean bathymetry and hydrography; sea as source of chemicals and minerals; sea as source of food, including fisheries; equipment and research.

472 — Ocean Engineering $65

Submarine geology and geophysics: undersea region as environment, habitat and sea bed resource; undersea chambers, construction methods, drilling and sampling, exploration, laboratories, ocean floor mining and research, underwater life-support systems and specialized equipment; use of diving and salvaging apparatus, submersibles and undersea vehicles and systems

480 — ENGINEERING GEOLOGY

481 — Geology and Geophysics $100

Engineering aspects of earth sciences including economic geology, geological dating, geomorphology, physical geology, regional geology, sedimentology, stratigraphy, structural geology and tectonics; factors affecting construction and location of engineering works due to geological conditions, geochemistry, geothermal phenomena, and terrestrial electricity, magnetism and physics including properties of ionosphere and upper atmosphere generally of geophysical interest.

482 — Mineralogy and Petrology $100

Chemical and physical properties, classification, composition, crystallography, formation, nature, occurrence, origin and use of minerals occurring naturally including precious and semi-precious gems, rocks and stones, lithology, petrography and petrology generally; regional mineralogy.

483 — Soil Mechanics and Foundations $100

Design and construction of foundations and soil structures related to engineering works such as buildings, dam sites, earthwork, embankments, and earth retaining structures; investigations and soil surveys by means of boreholes, sampling and other techniques, properties of clay, gravel, muskeg, permafrost, sand and silt; grouting, soil compaction, consolidation and stabilization, testing and evaluation of such mechanical and physical properties as bearing capacity, permeability, strength, and trafficability.

484 — Seismology $65

Analysis, recording and study of earthquakes, microseisms and other seismic action due to earth disturbances and volcanic eruptions, design of earthquake resistant structures; landslides, tsunamis and other secondary effects of earthquakes, seismic stations, seismographs and seismometry.

CLUSTERING ALGORITHMS

The mathematical steps required to construct clusters are simple. One way to do it is to define the distance between all pairs of records by the equation:

$$D(Ri,Rj) = 1 - \frac{N(Ri \cap Rj)}{N(Ri \cup Rj)}$$

Where: $D(Ri,Rj)$ = Distance between records i and j.
$N(Ri \cap Rj)$ = The number of terms <u>in common</u> between records i and j.
$N(Ri \cup Rj)$ = The number of terms <u>in either</u> i or j.

This distance is known as the Tanimoto or Jaccard distance.[22] Clearly, this equation satisfies the intuitive notion of distance. If records i and j have all their terms in common, the distance between them is zero. If records i and j have no terms in common, the distance between them is the maximum, 1. Thus the distance between records is just a measure of the term overlap between them.

One possible procedure for using the distance measure to partition the retrieval set is to find the distances between all pairs of records, and then to join into clusters those records that are separated by the smallest distances. That is, join the closest pair, then the next closest pair, etc., until only a manageable number of groups, about 20, remain. Many variations on this theme have been tried by various research groups.[22]

All experiments in this study were performed using a variation of this procedure called the Lance and Williams "Group Average" algorithm.[23,24] This selection was based on several factors. First, since the clustering was only to be applied to small files, algorithms that depend on $N_F{}^2$ instead of the less expensive $N_F \ln N_F$ in their space and time requirements could be afforded. Second, the Lance and Williams algorithm can readily be modified to accept distance thresholds, statistical term weighting and multi-stage processing. Following Van Rijsbergen[25], it has been found that most measures yield nearly equivalent results since they use the same information. The steps to the algorithm are:

1. Calculate the distances between each pair of records.
2. Select the two closest entities (either single records or clusters) and merge them to form a new cluster.
3. Calculate the distance from the new cluster to each remaining entity.
4. If more than one entity is left, go back to 2.

The calculation in Step 3 is as follows:

If record i and record j have been merged to form entity x, and the distance between record i and record j is denoted D(Ri,Rj), then for all entities q,

$$D(q,x) = \frac{N(Ri)\cdot D(Ri,q) + N(Rj)\cdot D(Rj,q)}{N(Ri) + N(Rj)}$$

where N(Ri) is the number of records in entity Ri, which is one. Similarly, N(Rj) = 1, and N(x) = N(Ri) + N(Rj) = 2.

This is, then, an agglomerative method. The clusters grow by fusion until the entire corpus forms one cluster. The corpus can the be divided into "Ø-clusters" by taking all the clusters farther apart than Ø. The distance between any two records can be defined as the distance at which those two records are first joined in one cluster.

The result of this sort of clustering is generally represented by a tree structure, called a dendrogram in which each record is represented by a leaf. Nodes in the dendrogram, representing joined records, are formed at characteristic distances. The distance between two records is the distance at which they are first joined (See Figure 6).

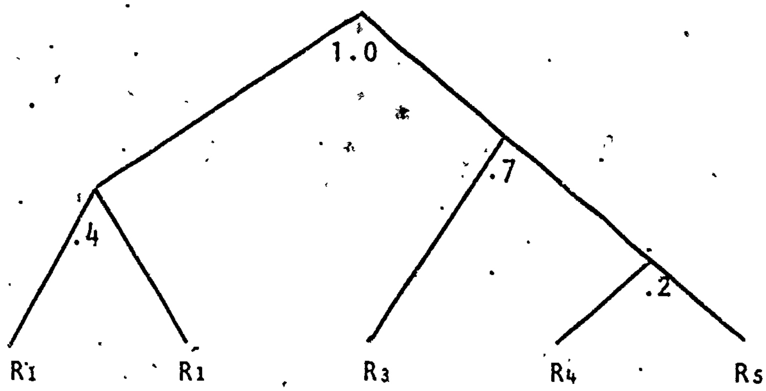
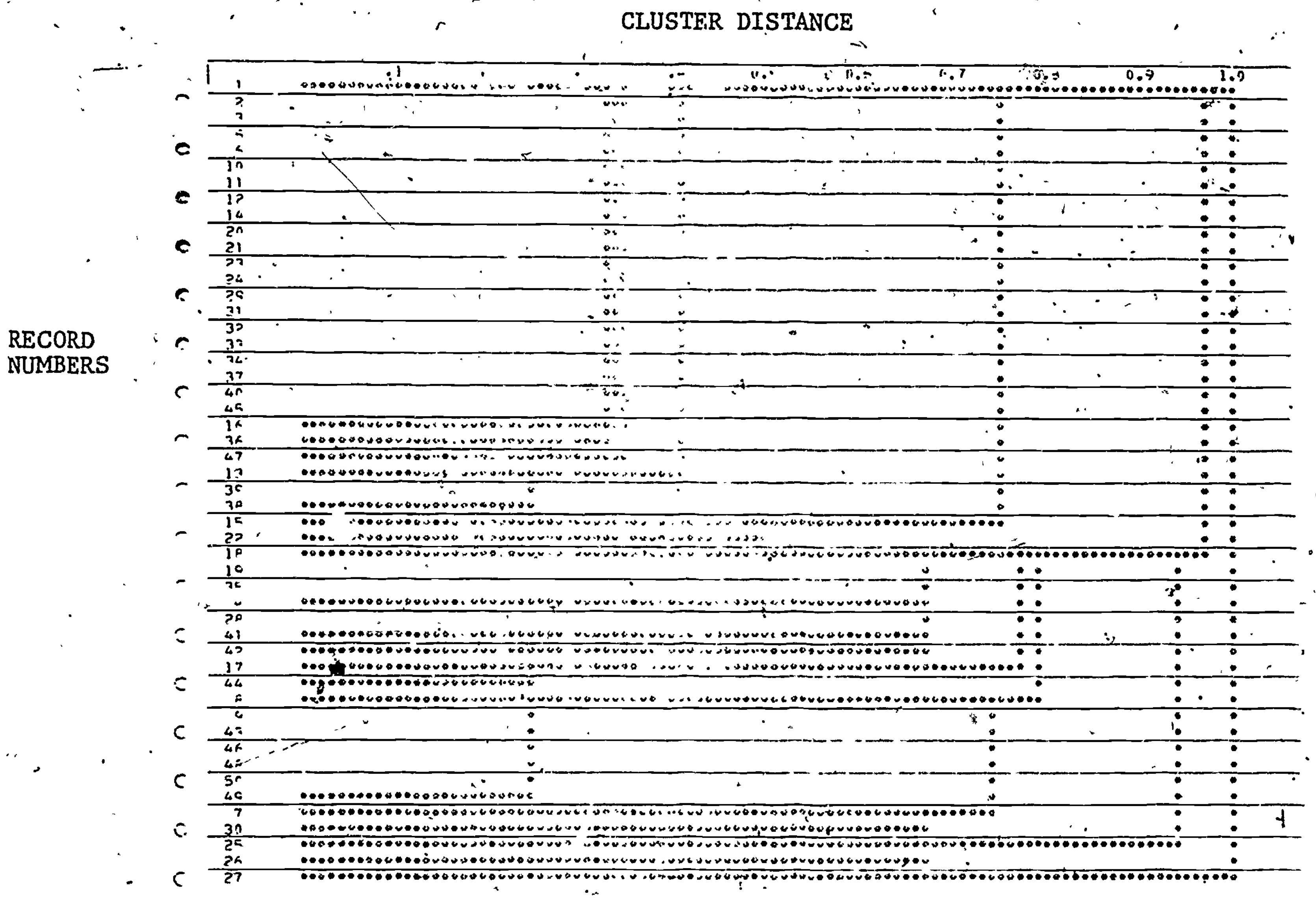


Figure 6. Prototype Dendrogram

In most dendrograms nodes will occur at several different distances between 1 and 0.

Lacking a plotting device, computer generated dendrogram representations had to be reformatted somewhat to be suitable for display (See Figure 7).

Though the previous discussion has been concerned with the clustering of records; it is often useful to cluster terms, thus building groups of "synonyms". This can be done using exactly the same algorithm as before. Just as a bibliographic record can be treated as a list of terms to be clustered, the inverted file of postings that is associated with a single term can be treated as a list to be clustered. The equivalence of those procedures is indicated graphically in Figure 8.



Note: for example, that records 13 and 39 are joined at a distance of .25. Similarly, records 3 and 5 are joined at a distance of .33, and so have less term overlap than do records 13 and 39.

Figure 7. Sample Dendogram

RECORD 1

Term 1
Term 7
Term 8
Term 26
Term 147

RECORD 2

Term 6
Term 8
Term 26
Term 35
Term 104

a. Record Clustering

TERM 1

Post 8
Post 17
Post 108
Post 110

TERM 2

Post 6
Post 7
Post 8
Post 14

b. Term Clustering

The same algorithm that clusters records over their terms (a) can cluster terms over their postings (b) (Links shown).

Figure 8. Relation Between Record and Term Clustering

MEASUREMENT OF CLUSTERING PARAMETERS

Three key parameters characterize the usefulness of a cluster run:

1. The fraction of the file that is allocated to groups (coverage),
2. The average size of the groups formed (agglomeration), and
3. The fraction of the file that is allocated correctly (accuracy).

These parameters are evaluated according to the following rules.

Coverage: Any record is counted as clustered at a distance D if it participates in at least one join with another record at any distance less than or equal to D.

Agglomeration: Agglomeration (N_A) is measured as the average size of the clusters that are formed at a distance D. It is calculated as the number of records clustered (N_C) divided by the number of clusters (N)

$$N_A = \frac{N_C}{N}$$

Accuracy: If records of two kinds (A and B) are clustered (at a distance D), a cluster is counted as being of the A type if the majority of records in the cluster are A type, and as B if they are of B type. The A records in an A cluster are counted as correct, and the B records in a B cluster are counted as correct. Conversely, A records in a B cluster, or B records in an A cluster, are counted as incorrect assignments. If there are an equal number of A and B records in a cluster, then half of the total are counted as correct.

3. EXPERIMENTS

In order to test the feasibility of the sequential processing hypothesis, 4 experiment were conducted. Each experiment was designed to answer a specific question about the limitations of statistical string processing.

EXPERIMENT 1

The question addressed by the first experiment is, "Can direct vocabulary feedback to a searcher act as a useful summary level device?" That is, in seeking a mechanism to characterize a retrieved set, it is natural to consider a sorted list of the terms present in the records. Current on-line systems provide some vocabulary support, such as listing terms present in the data base that are alphabetically close to a given term or related to a given term by subject content (broader term, narrower term, synonym, etc.)[26]. However, the information given by this vocabulary support capability applies to an entire data base, rather than to a retri ved set. That is, one can readily obtain a sorted list of the terms present in the whole data base, but not the terms in a retrieved set.

Since the searcher evaluates records by looking for occurrences of terms, it seems natural to have the computer simplify the task by presenting to the user a sorted list of the terms present in the initial retrieved set. In this experiment, it was found that the number of terms on which the relevancy decisions are based is usually just a few percent of those terms present (though the set of crucial terms may be different for different users even if they are concerned with the same initial retrieved set). Thus, it is appealing to consider how the terms might be sorted for feedback. Some sorting is necessary, as even for a mere 100 records there are about 1,000 unique terms in the title and index field for CACon - too many for the user to benefit from having to scan all of them rather than the entire records. It was conjectured that simple frequency criteria might be sufficient to identify the key terms. For this experiment, typical retrieved sets, containing relevant and non-relevant records (about 50 of each type) were characterized by the terms that they contained. The crucial terms, on which the relevancy decisions were based, were identified. It was found that they could not be identified by simple statistical criteria. Often, low frequency terms were crucial when they indicated specific concepts that were not relevant. However, in other cases, high frequency terms were necessary. The inability of gross frequency data to select terms appropriate for searcher feedback led to the postponement of consideration of vocabulary feedback until after vocabulary mapping experiments had been completed (Experiment 4). The vocabulary mapping involved semantic input and promised to increase the efficiency of retrieval above the level of purely frequency-based criteria. The possibilities of vocabulary feedback based on this semantic input instead of gross frequency data are discussed further in Section 3.

EXPERIMENT 2

The following questions were addressed by the second experiment. Can clustering resolve record classes with substantial vocabulary overlap such as will occur as the result of a Boolean retrieval? How does the resolution depend on the mathematical details of the clustering procedure? What are the relative contributions of the various record fields (title, index, abstract, CODEN) to resolution? That is, clustering can be expected to easily resolve records from disparate disciplines into separate groups, in cases where overlap between the two disciplines is small, such as high temperature physics and botany. It is less clear that clustering can successfully resolve records from disciplines with much vocabulary overlap. (See Figure 9).

The design of Experiment 2 is indicated in Figure 10. Fifty records were taken from each of two sections of CACon or COMPENDEX and were put into one file of 100 records. CACon has a subject organization, so that all the records contained in a given section pertain to a given subject, such as "Hormone Pharmacology" or "Mammalian Biochemistry". Card-Alert-Codes play a similar role in COMPENDEX. When the file with 100 records is clustered, ideally it would divide into two clusters, each containing 50 records from one section.

Some typical results are shown in Figures 11 and 12. When the two sections used are disparate in subject area, such as the sections on "General Biochemistry" and on "Terpenes", the separation achieved closely approximates the ideal when the title and index fields are included.

When the two sections selected have greater vocabulary overlap, such as the sections "Terpenes" and "Carbohydrates", the separation is much less successful. A number of generalizations can be drawn from the data. In an effort to measure the effect of the mathematical details on the separation, several different clustering procedures were tried. In general, it was found that the problem lies mostly in the structure of language, not in the mathematics of classification. That is, the experiments suggest agreement with Van Rijsbergen[25], that most measures yield similar results because, ultimately, they are based on the same information. Also, it seems that further improvement requires additional preprocessing, such as generation of a degree of semantic structure for the vocabulary. Clustering without any additional vocabulary preprocessing will be called simple clustering.

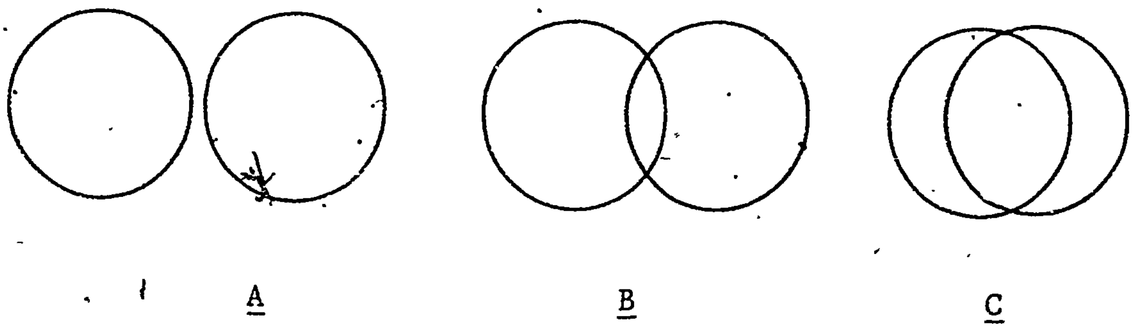


Figure 9. Effect of Term Overlap on the Resolution of Record Groups

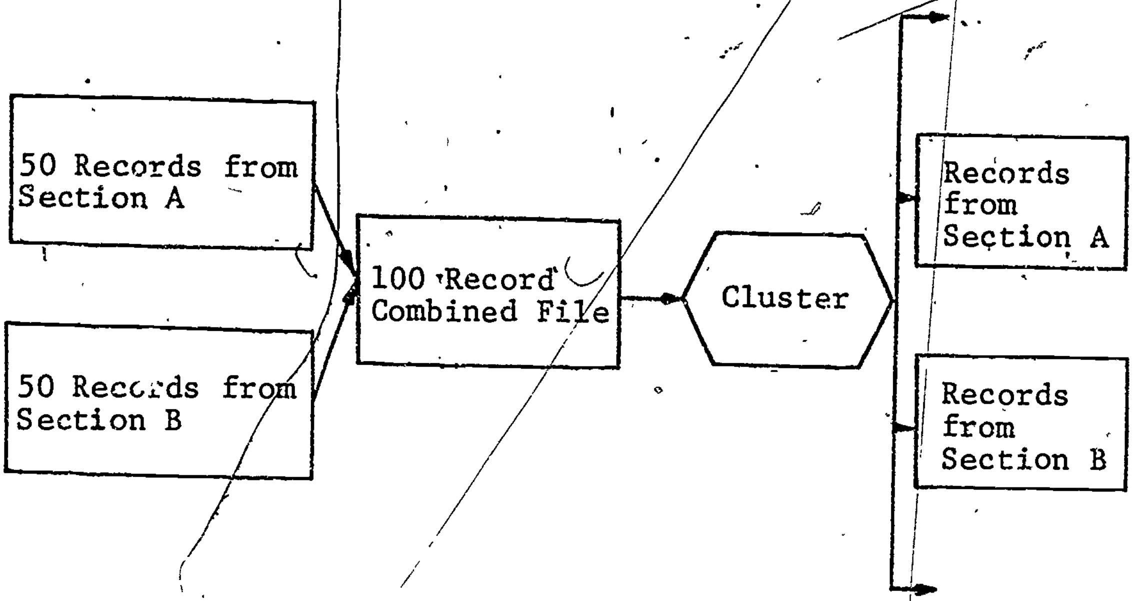
To be useful as a second step retrieval device, clustering must function well in Case C.

Clearly, most algorithms can separate groups such as A wherein term overlap is negligible. Separation is more difficult for B, wherein term overlap is slight but non-negligible. Separation for Case C is required if clustering is to be successful as a second step search mechanism, for the set selected by the first search step will have much overlap, as all members were selected by a search strategy.

The results of Experiment 2 suggest that records from different supersections of CACon are like case A and are easily separated. Most records from two different sections are like case B and are separated with acceptable efficiency. However, records from related sections are like case C and are not separated acceptably by simple clustering. Since case C corresponds to the kind of overlap found for record sets retrieved by a Boolean search, it seems unlikely that simple clustering can partition retrieved search sets into relevant and non-relevant clusters with acceptable efficiency.

The surprising result that the inclusion of the abstracts field made only a small contribution to record resolution by simple clustering is related to the effect of high frequency terms on the pattern, and is discussed in Section 4.

Experiment 2



Results:

- Effect of variation in cluster algorithm
  - Using only non-singular terms improves cluster separation
  - Details of the distance measure seem to have only a small effect on the partition
- Effect of different data fields on the partition
  - CODEN field is useful
  - Index field is the best
  - Title fields is second best
  - Abstract field makes only a small contribution
- Effect of section choice on accuracy of partition
  - Records from sections characterized by very different vocabularies are easily distinguished
  - Records from sections characterized by similar vocabularies are not easily distinguished

Figure 10. Design and Conclustions for Experiment 2

Typical Results: CACon Sections on General Biochemistry and on Terpenes

Field	Records Clustered	Records Clustered Correctly	Number of Clusters
IDEAL	100	100	2
T	85	61	5
I	89	84	9
C	60	58	16
T + C	93	86	4
T + I	100	98	2
T + C + I	100	98	2

Results for CACon Sections on "Terpenes and "Carbohydrates"

T	84	53	12
T + C + I	96	73	8

T = Title

I = Index

C = CODEN

Figure 11. Typical Results for Experiment 2

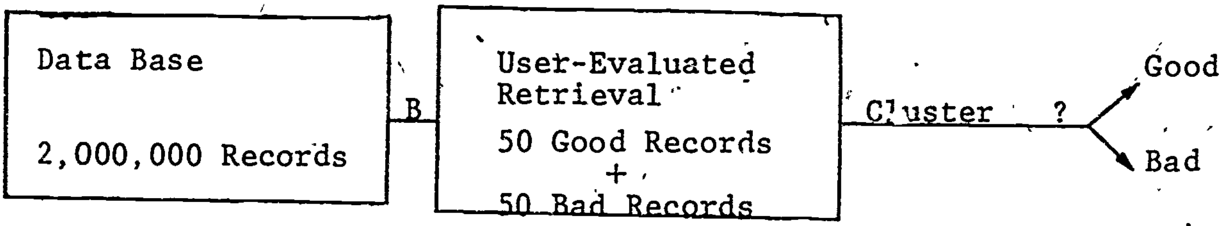
FILES	FIELDS	INCLUDING SINGULAR TERMS	COVERAGE	ACCURACY	NUMBER OF CLUSTERS	DISTANCE
CA6 & CA30	5	No	89	84	9	.98
CA6 & CA30	1,2	No	93	86	4	.99
CA6 & CA30	1,2	Yes	92	85	16	.99
CA6 & CA30	1,2,5,Sect #	No	100	98	2	.99
CA6 & CA30	1,2	No	100	89	4	.99+
CA6 & CA30	2	No	86	77	6	.99
CA6 & CA30	2	Yes	83	74	17	.97
CA6 & CA30	2,5	No	100	98	2	.98
CA6 & CA30	1,2,5	No	100	97	4	.99
CA6 & CA30	2,5,Sect. #	Yes	100	100	5	.99
CA30 & CA33	5	No	76	57	11	.99
CA30 & CA33	1,2,5	No	96	73	8	.99
CA6 & CA33	1,2	Yes	92	85	16	.99
CA6 & CA8	2,5	No	100	71	5	.90
CA6 & CA8	2	Yes	79	67	16	.99
CA6 & CA8	1,2	No	95	72	5	.99
CA6 & CA8	1,2,5	No	100	96	3	.90
E1452 & 817	2	No	100	81	7	.99
E1452 & 817	2,5	No	100	100	3	.98
E1452 & 453	2,5	No	80	57	2	.90
CA8 & CA74	2,5	No	100	95	5	.98
CA36 & E1815	2,5	Yes	100	82	3	.98
E1535 & 537	1,2,5	No	86	71	4	.95
E1535 & 537	9	No	100	54	2	.99
E1461 & 535	1,2,5	No	100	98	4	.98
E1461 & 535	9	No	100	58	2	.86
E1452 & 453	9	No	60	49	11	.95
E1453 & 461	1,2,5	No	100	93	3	.93
E1453 & 461	2,5	No	100	90	3	.97
E1452 & CA8	2,5	No	100	100	3	.96

Figure 12. Experiment 2 - General Summary of Data

EXPERIMENT 3

The question addressed by the third experiment is, "Can simple clustering separate the user-judged relevant records from the non-relevant ones?" The experimental procedure is illustrated in Figure 13. Searches performed by IITRI's Computer Search Center and evaluated by users in the normal course of center operations were used as the basis of the test. For each of the experimental tests, fifty relevant and fifty non-relevant records, for one user, were put together in one file of 100 records. Then, the file was clustered. Again, as in Experiment 2, the ideal condition would be to have two clusters formed, one with 50 relevant records and the other with 50 non-relevant records. Results indicate that although the separation produced by simple clustering is not good enough for it to serve as a reliable high-precision second step mechanism, it does approach an acceptible level in many runs. Hence, motivation was high to explore the structure of vocabulary and its implications in the fourth experiment, in the hope that the addition of some semantic information would increase the second step efficiency to the point that it would be immediately practical.

Experiment 3



Results:

- Clustering assignments are made with good accuracy at small cluster distances but not at large ones.
- The fraction of the file that is clustered is sufficient only at large cluster distances.
- The average cluster size is acceptible only at very large cluster distances.
- Simple clustering is not practical as a second-step mechanism for any file configuration tried, although results approach practical levels for many individual runs.
- Further progress would be greatly aided by incorporation of a degree of semantic information in the clustering process.

Figure 13. Experimental Design for Experiment 3.

Three key parameters specify the usefulness of a cluster run, namely coverage, accuracy and agglomeration. If coverage is low, part of the file is not included in the pattern; if accuracy is low, the pattern is worthless; if agglomeration is low, the number of decisions that the user saves is low. That is, if there are N_A records per cluster, and only one of them need be evaluated to evaluate all by implication, then (N_A-1) decisions are saved per cluster. If a file of N_F records is divided into groups of size N_A, then there are N_F/N_A groups and the total number of decisions is reduced from N_F to N_F/N_A. Unless N_A is large, the savings is small. Figures 14, 15 and 16 show the summary of these three parameters obtained, as a function of cluster distance for 50 user evaluated retrievals (each containing 50 relevant plus 50 non-relevant records) clustered under the protocol of Experiment 3. Each data point represents the average value of a parameter for the 50 runs, and each vertical bar delimits the one standard deviation interval from the average at that point. According to Figure 14, only at distances greater than 0.95 (about 1 overlapping term among two records with 10 terms each) is substantially all of the file clustered. About 80% of the file is clustered at a distance of 0.8.

According to Figure 15, the number of records clustered correctly is approximately equal to the number clustered at small distances, but it falls off at high distances. At a distance of about .95, only about 70% of the records are clustered correctly. According to Figure 16, the agglomeration does not become appreciable until cluster distances are greater than about 0.9. In summary, simple clustering can separate relevant records from non-relevant ones with sufficient accuracy only at very small distances, whereas agglomeration and coverage are sufficient only at large distances. To improve upon this situation it was decided, upon surveying individual runs for the reasons of clustering failure, that a mechanism was needed to allow the relation of non-identical strings on the basis of their semantic relationships. To that end, the vocabulary mapping experiments were initiated.

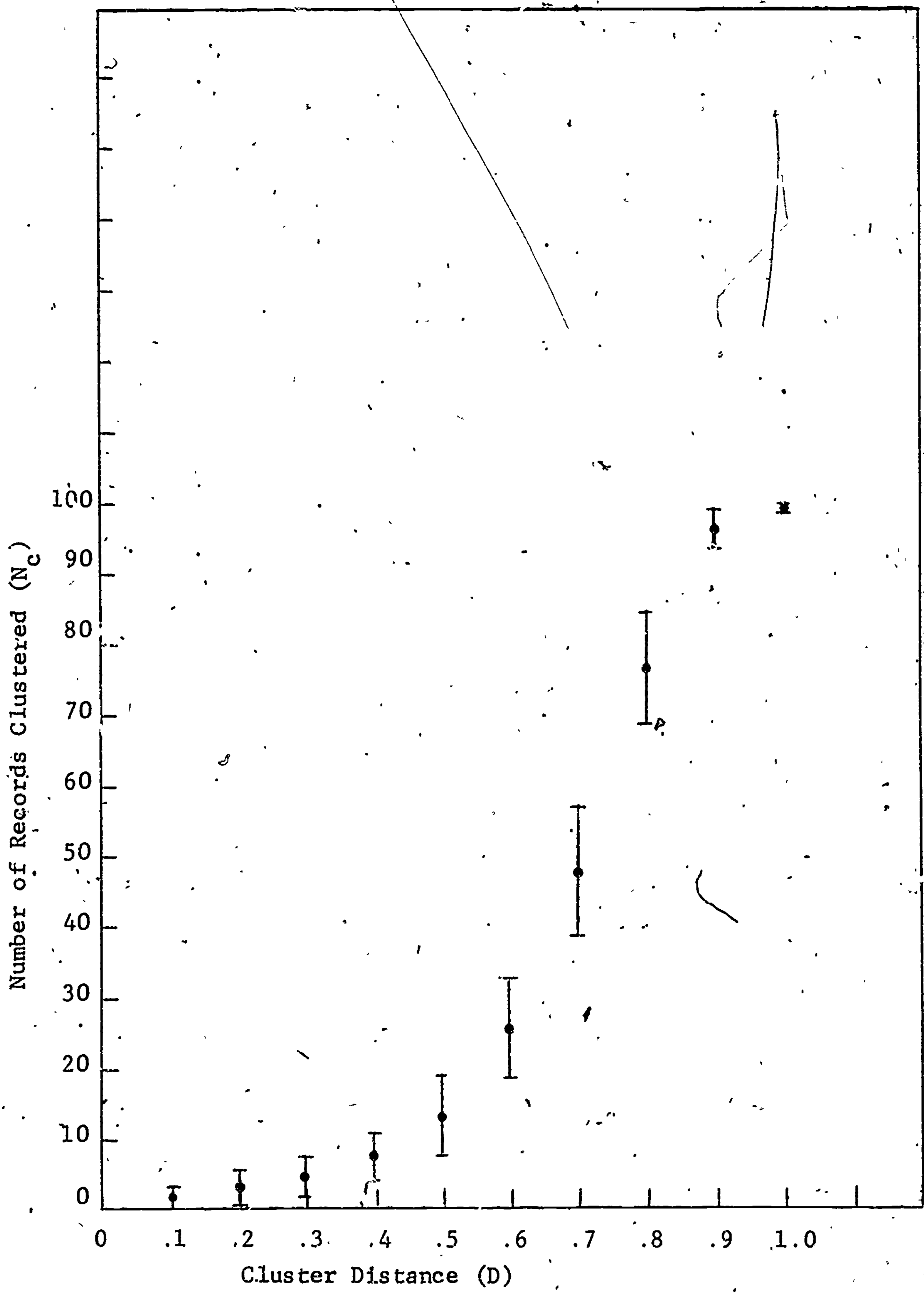


Figure 14. Number of Records Clustered vs Cluster Distance

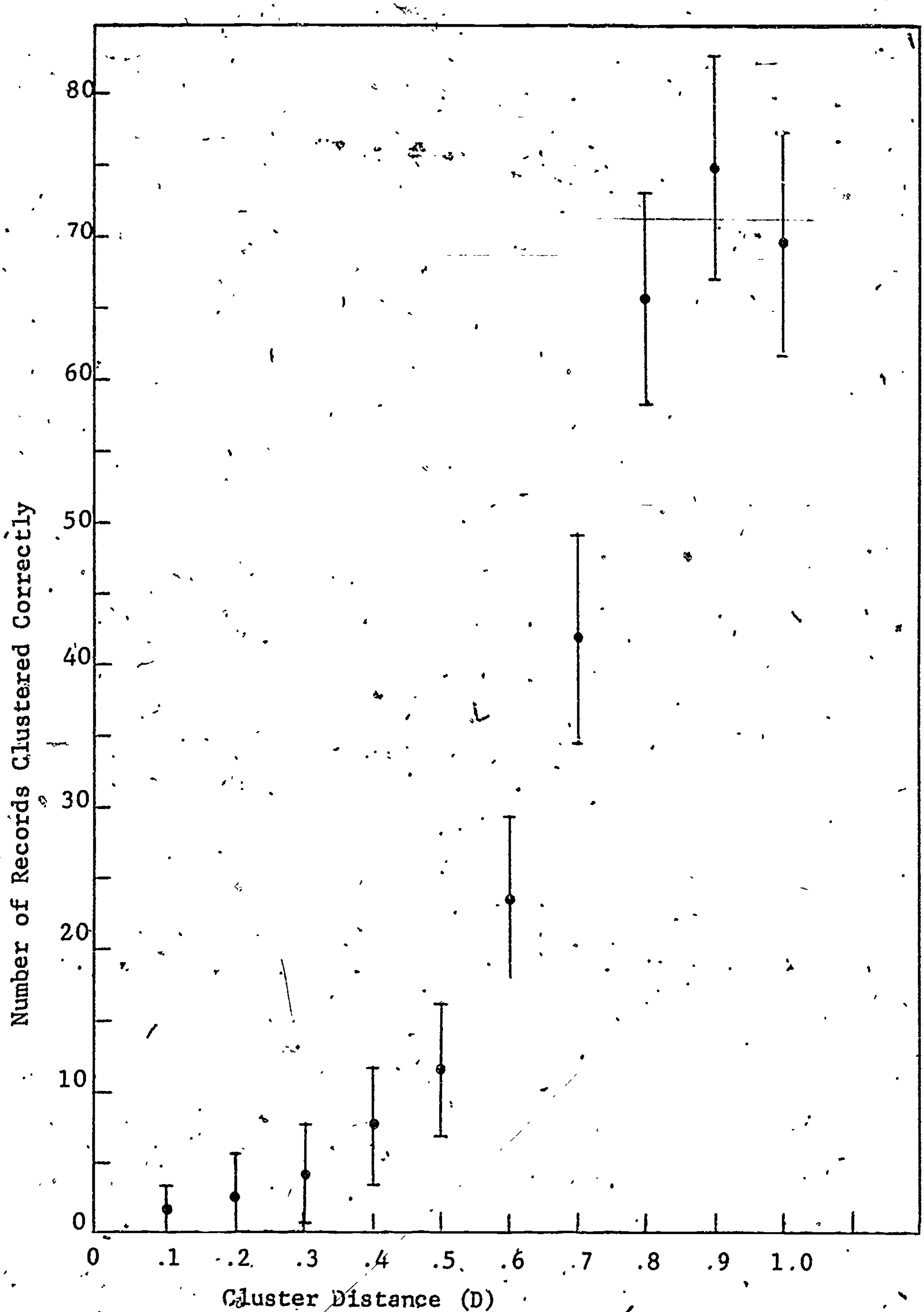


Figure 15. Number of Records Clustered Correctl. vs Cluster Distance

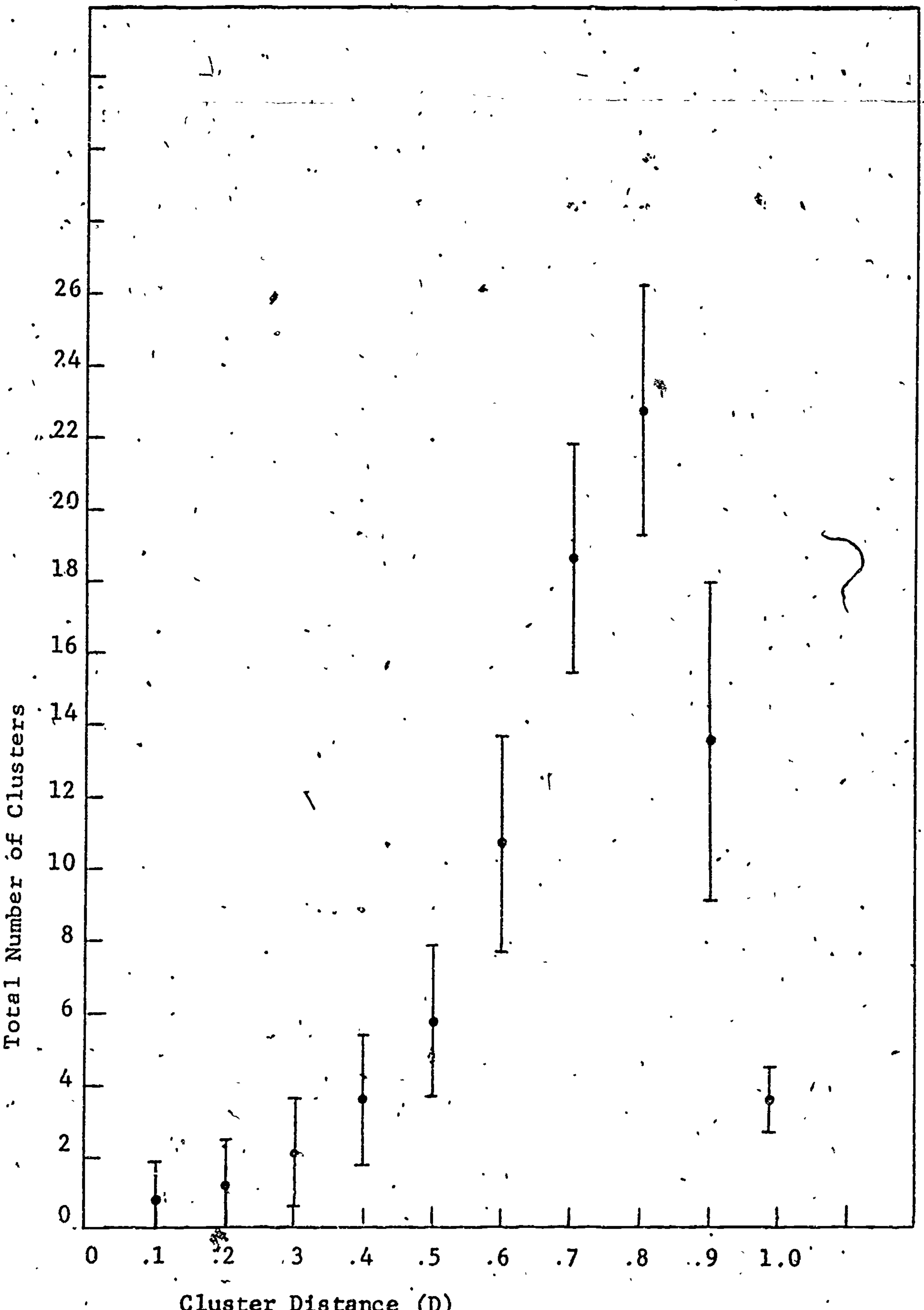


Figure 16. Total Number of Clusters vs Cluster Distance

EXPERIMENT 4 - VOCABULARY MAPPING

Even before the first 3 experiments were done, it was recognized that there is one major reason why simple clustering would not be expected to work well enough to correctly classify a collection of records with significant vocabulary overlap. Any collection of records can be classified (ordered or partitioned) in many different intellectual ways. Simple clustering, as described earlier, is merely one arbitrary way of classification. As such, it is not clear that it should be expected to separate the relevant records from the non-relevant ones or to separate records into groups that are meaningful to a given user because what is relevant depends on the intellectual classification principles of the user. For example, suppose that the user entered a Boolean search on the subject of plants and air pollution. The resulting retrieval could be intellectually categorized according to the species of plants involved, putting, for instance, hardwood trees in one group, softwood trees in another, shrubs in another, etc. Alternatively, the records could be intellectually categorized according to the chemical air pollutants involved, SO_2 in one group, NO_2 in another, ozone in another, etc. Similarly, the intellectual categorization could be based on weather conditions, geography, economic impact, country of origin, etc. Thus, since the computer at present does not have the definitions of the terms, the problem of constructing user-meaningful partitions has two levels. First, the system has to have a way of homing in on the intellectual principle of classification (i.e. in the sense that categorizing the example retrieval on the names of the plants involved is an intellectual principle of classification). Second, a way has to be found to direct the classification mechanism (clustering) to use the classifying principles specified by the user.

The solution of these problems requires that the system has additional semantic information available. That is, while the full dictionary-type definition of each string may not be required for processing at this stage, there must be at least enough information to distinguish the terms among the various common intellectual organizing principles to which they may apply. To this end, it has been found desirable to map each term into a conceptual category. Thus, for instance, suppose oak were mapped into the category "plant", NO_2 and SO_2 were mapped into the category "air pollutant", etc. Then the selection of an intellectual principle of classification would correspond merely to the selection of a term category. That is, if the terms that denote the names of plants were labeled as belonging to the class of plant names, they would be singled out by the computer as the string symbols on which to base a record classification even though the computer could not distinguish among those names any secondary characteristics (i.e., "tomato" is defined only as a member or the class of plant names). So the key is that to classify the records according to the principle "plant names", one should cluster on only the subset of all the terms present that pertain to plants. More generally, to classify records according to an intellectual principle, cluster on only the terms that are members of the term class that corresponds to that principle. Since the terms so chosen are only a small subset of all those that are present in a record, IITRI has named this process Subset Based Clustering, or SBC.

A secondary advantage of constructing term classes is that it offers the possibility of overcoming some of the limitation of the binary value of string match. For example, the term "dog" and the term "greyhound" are not identical character strings, and so they do not match. Similarly, the terms "bean" and "dog" do not match. Yet, clearly "dog" is much more similar to "greyhound" than it is to "bean". One way to enable the system to compute on the basis of degrees of similarity is to record the term association probabilities for a body of text, and to make the assumption that terms that tend to occur together are semantically related. This technique has been used to great advantage by Salton[17]. Unfortunately, it is expensive to compute, store and access term correlation coefficients for large data bases. This project has attempted a different approach based on the definition of intellectual word classes.

One might argue that terms are defined by the context in which they occur. That is, medical terms occur in medical records, engineering terms in engineering records, etc. Using this idea, one might represent each term by the list of records in which it occurs. An initial attempt to overcome the limitation of binary match (matching is either identical or zero (1 or 0)) was based on this concept. The idea was to take the small record set that would result from a Boolean search, and to cluster the terms over the records, in effect defining a similarity between terms based on their co-occurrence within the records of the small Boolean search. Then, the term similarities would be used to cluster the records (sequence shown in Figure 17). A typical term map resulting from such a sequence of operations is shown in Figure 18. This sequence of operations is appealing because it is inexpensive and self-contained. The clustering of terms involves only the small set and requires no dictionary loop-ups. Unfortunately, it was found that this processing sequence makes only a marginal improvement to the resolution of record clusters. The essential

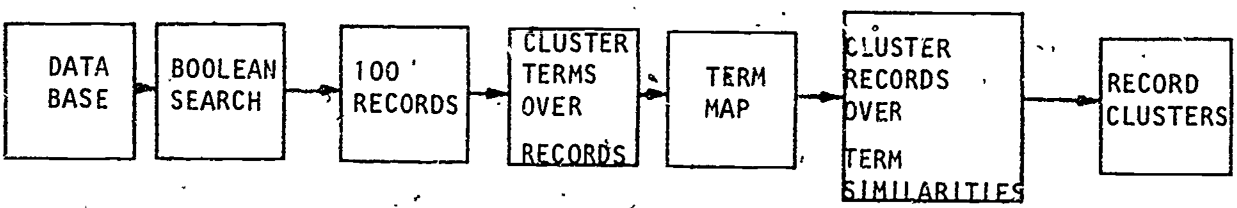


Figure 17. Retrieval Procedure Using Record and Term Clustering (Preece Algorithm[27])

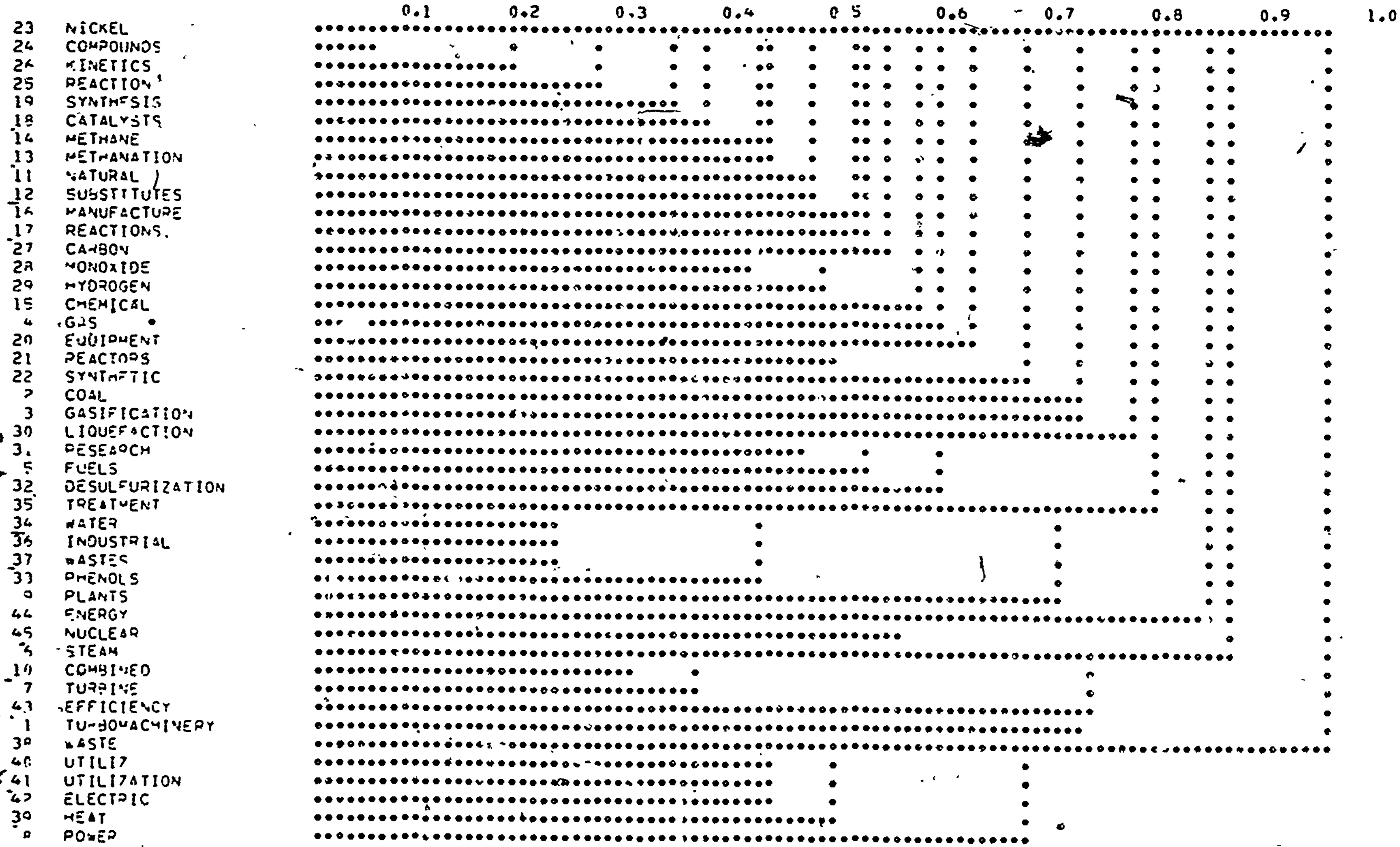


Figure 18. Typical Term Map Derived by Procedure of Figure 17

problem is that defining similarities between terms is essentially a global property, and it is unrealistic to hope that the strings can be classified merely on the basis of their associations in a small record set. That is, similarities are not well enough defined using this method for small record sets, and for large record sets the process is expensive.

The process of context definition seemed to be sound, so additional effort was made to apply it on a global scale (i.e. to a whole data base). The conceptual organization of the CACon data base into supersections, sections and subsections (see Section 3) suggested that terms could be characterized by their occurrence in this hierarchy. That is, records are filed by CAS indexers within the CACon section structure according to their intellectual content. Because the intellectual content is represented by terms, they are implicitly filing the terms according to their intellectual relationships. Accordingly, it should be possible to recover the mapping of the terms into the categories (sections) merely by counting the number of times that a term occurs in each of the sections, taking into account the fact that the sections have different overall numbers of records (and hence probabilities that any term will occur in a section), and looking for peaks in the distribution. When this is done for a typical term, using the 80 CA sections, the result is a plot such as that shown in Figure 19. Terms that occurred mostly in one section, like Term A, are characterized by the subject of that section. For instance, the term "estradiol", which is the name of a hormone, occurred almost exclusively in the section on "Hormone Pharmacology" (Figure 20). Hence, independent of any use of its dictionary definition, "estradiol" was identified as a hormone pharmacology type word. Other words like Term B, have a broader distribution but are still restricted to a limited range of sections, such as those relevant to organic chemistry, inorganic chemistry, etc. An example of this type of behavior is the term "fiber", which, as shown in Figure 21, occurred mainly in the sections on "polymer Chemistry". Other terms, such as "acid", Figure 22,

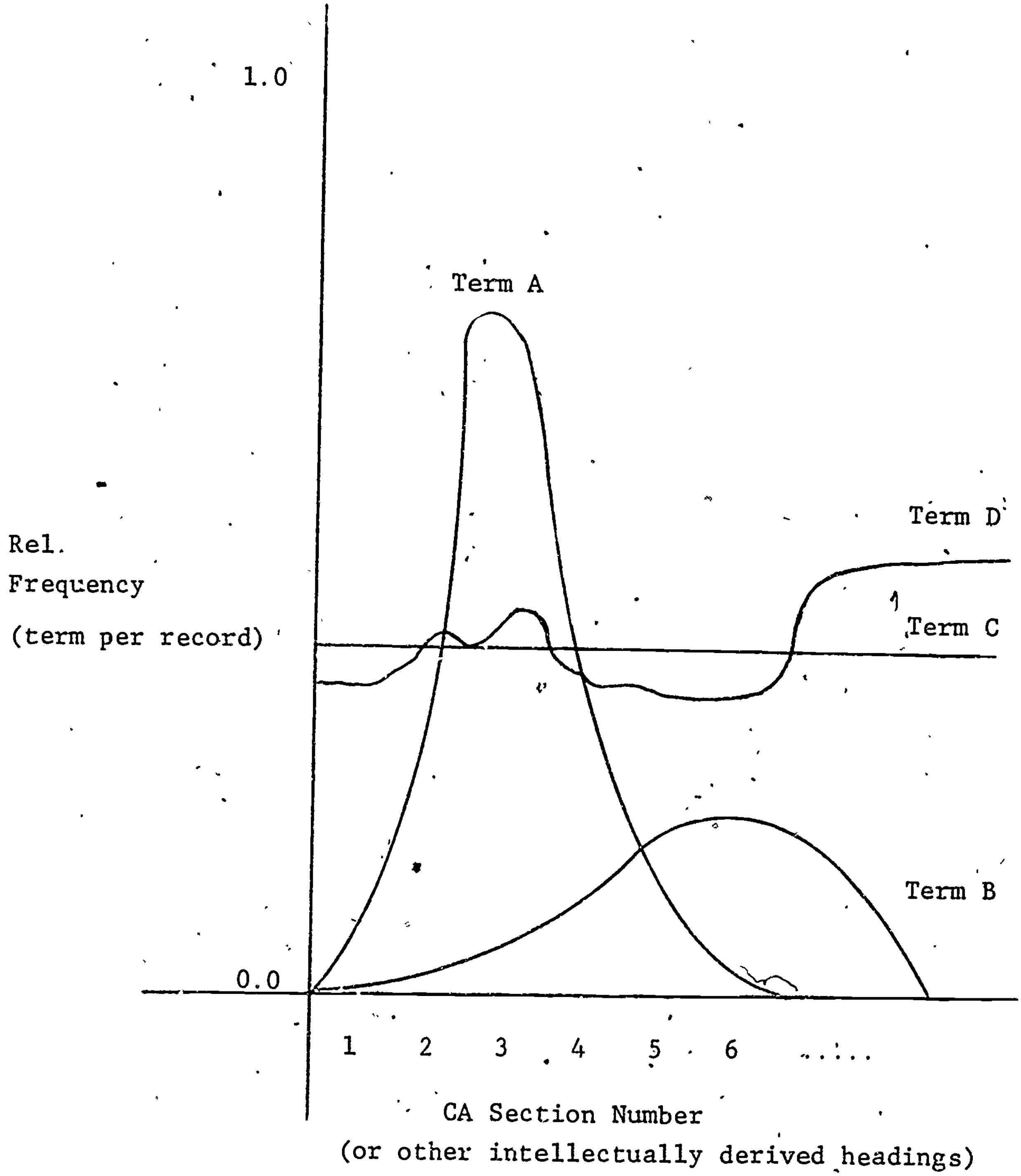


Figure 19. The Relative Frequencies of Four Hypothetical Terms in Each of the 80 CACon Sections

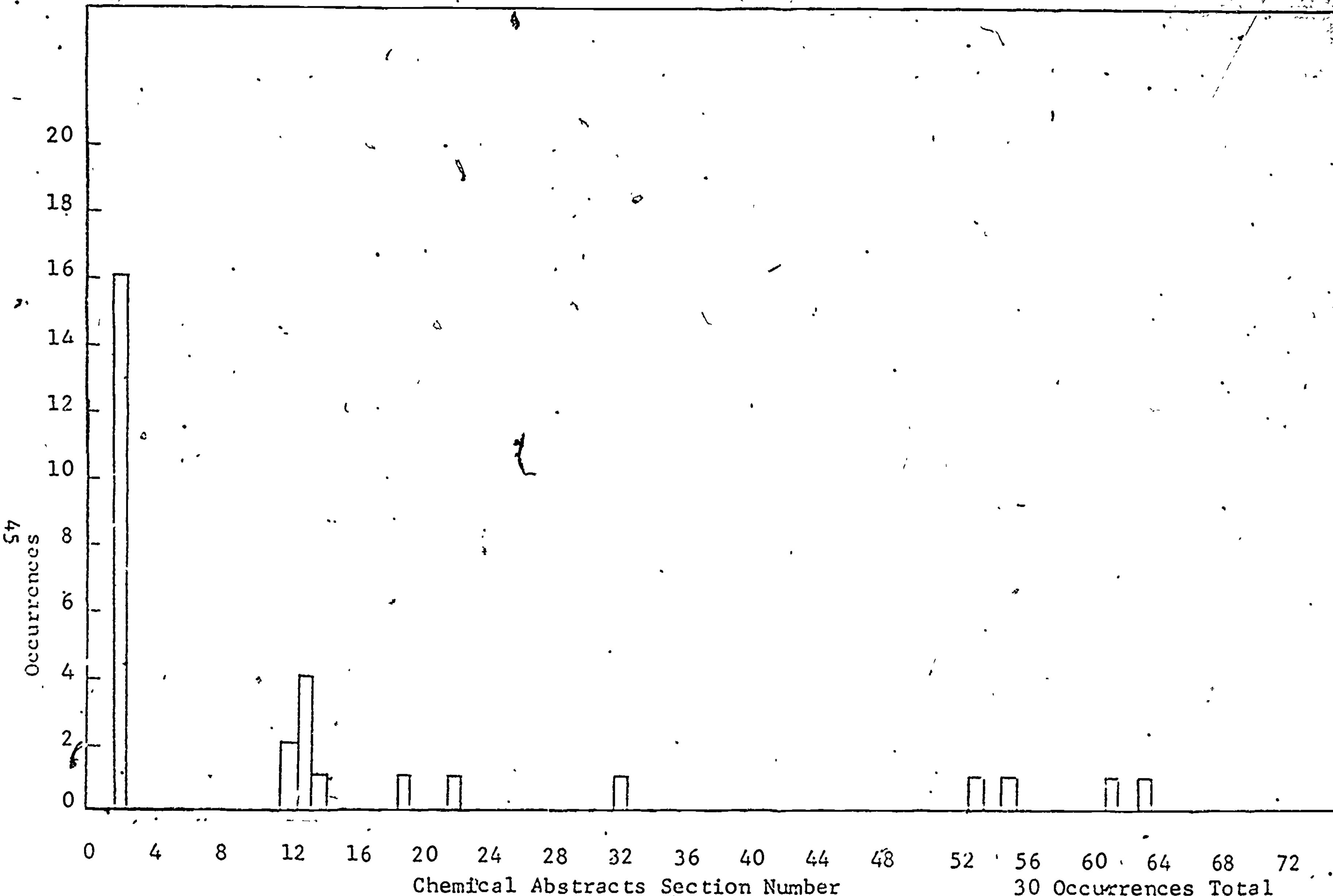


Figure 20. Distribution of the Term "ESTRADIOL" in CACon.

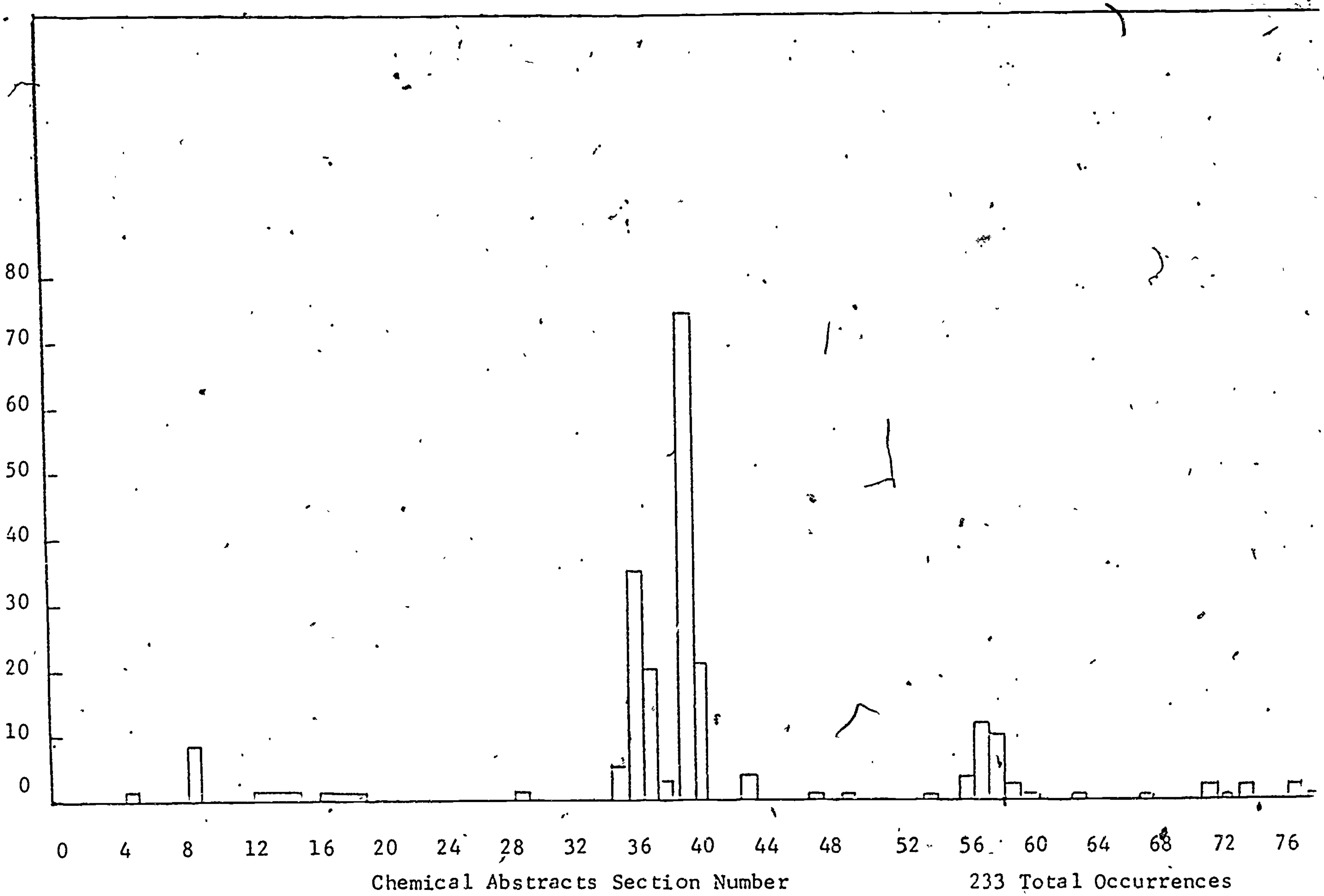


Figure 21. Distribution of the Term "FIBER" in CACon

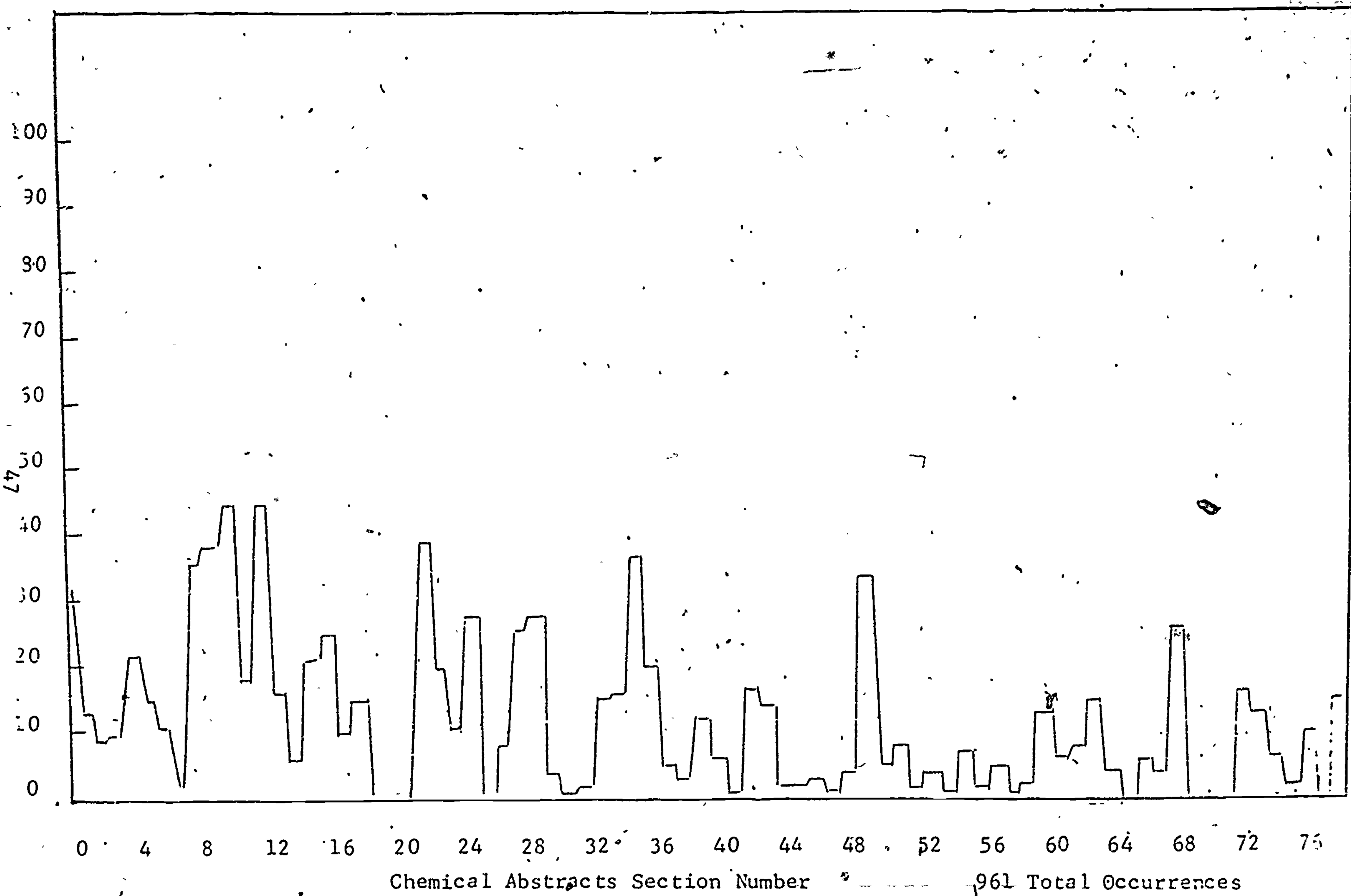


Figure 22. Distribution of the Term "ACID" in CACon

or "pressure" have distributions like terms C or D on Figure 19. The meaning of such flat distributions is that the terms are equally applicable to the concepts of each of the CAS Chemical Abstracts sections. This need not mean that C or D terms are not good discriminating words. Rather, it just means that their discrimination value is very limited with respect to the term classes consisting of CACon section labels. For instance, a term related to temperature or pressure may be of conceptual value for retrieval and may occur in only a small fraction of records. Still, if its distribution is flat, i.e. if it occurs equally in all CACon sections, then it cannot be assigned to a CACon section term class. The major advantages of this form of term classification are that the term classes and their headings are based on intellectual judgements. That is, records (and the terms they contain) are assigned to sections by indexers according to their record meaning. That is, indexers assign records to sections according to the meaning of the section title and terms. Further examples of word distributions are shown in Figures 22, 23 and 24.

Examination of the distributions of all the terms in two issues of CACon shows that most of the terms map easily into either a single section or a small group of sections. Some terms, such as "absorption", map into two sections or groups of sections, because they can have two separate meanings, as in the sense of physical absorption versus spectral absorption.

To characterize the degree to which the free text terms of CACon map into section or supersections, the distribution such as those thown in Figures 19 to 24, was generated for each test term. Then the fraction of normalized occurrences of a single term that occurred in the peak section of the distribution was calculated according to

$$f_1 = \frac{\text{the number of occurrences of a term in its peak section}}{\text{the total number of occurrences of the term}}$$

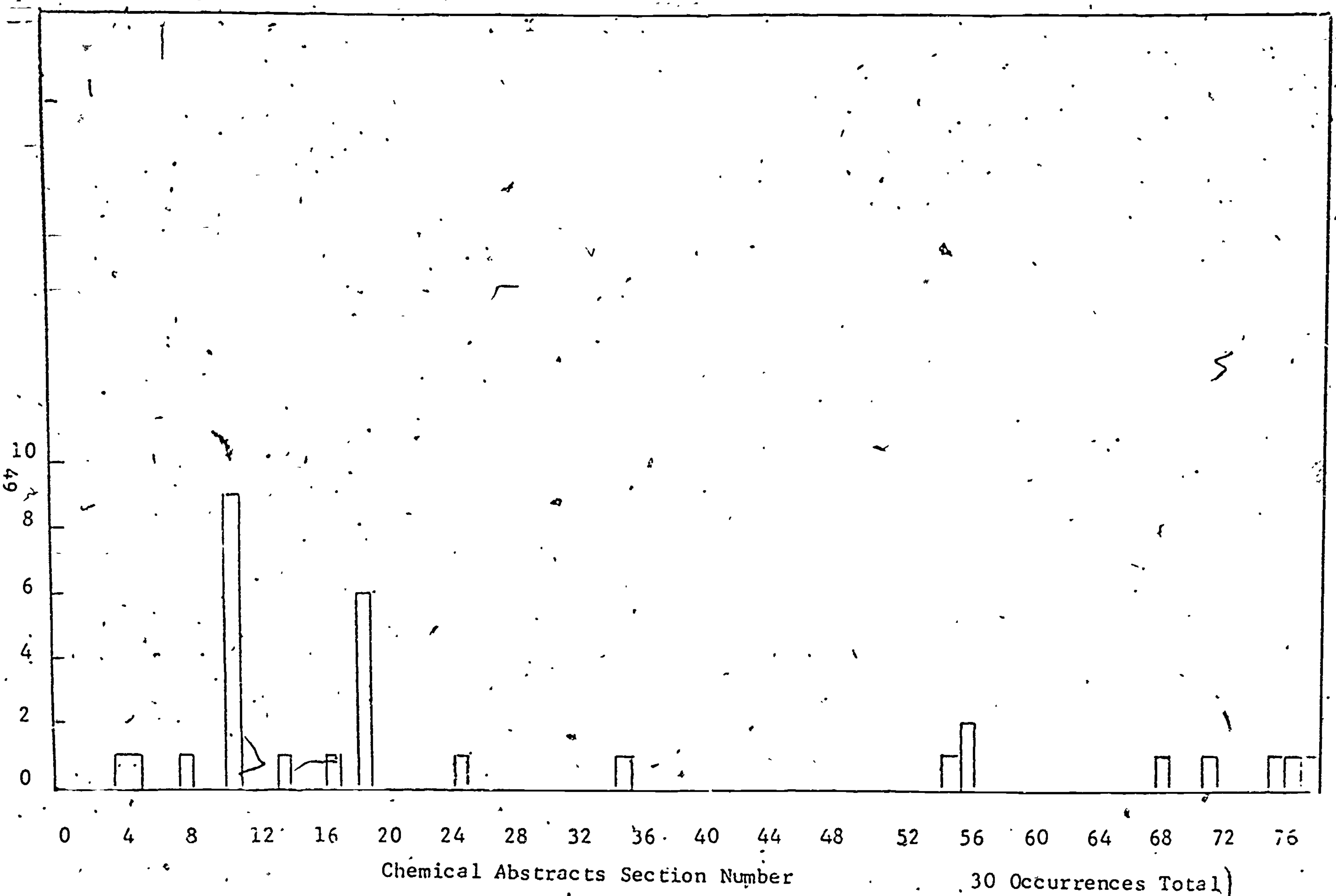


Figure 23. Distribution of the Term "PEA" in CACon

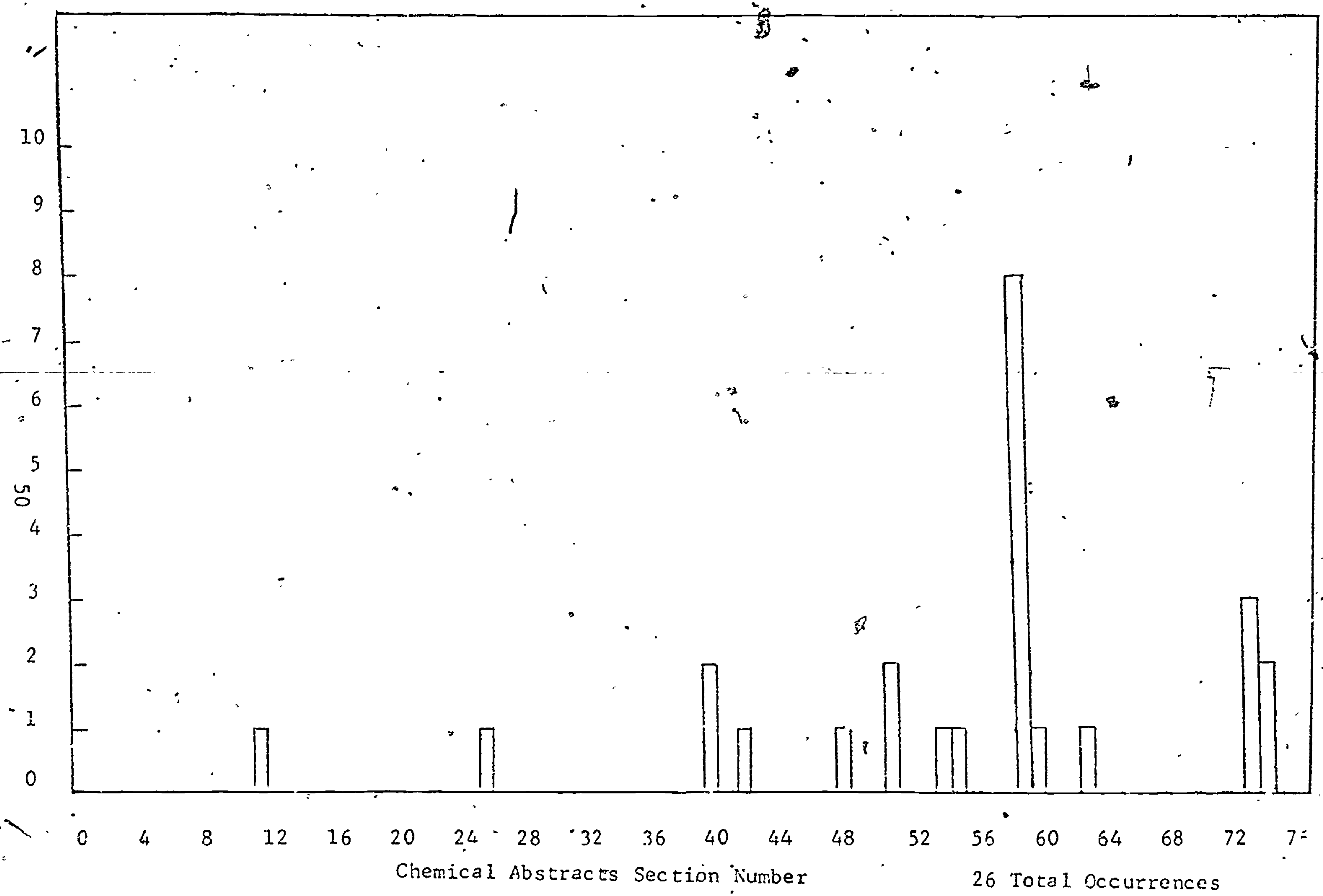


Figure 24. Distribution of the Term "FLOUROENYL" in CACon

The term counts are normalized to account for the fact that different sections contain different numbers of records. Similarly, the fraction of normalized occurrences of a term that occurred in the section with the second greatest concentration of that term was calculated according to:

$$f_2 = \frac{\text{the number of occurrences of a term in its second peak section}}{\text{the total number of occurrences of the term}}$$

The fractions f_1 and f_2 have the following properties. If a term occurs only once in the record set, $f_1=1$ and $f_2=0$. That is, if a term appears only once, then it can appear in only one section and so it must map into one section perfectly ($f_1=1$) and into no other ($f_2=0$). If a term occurs only twice, then $f_1+f_2=1$, since the term can occur in only two sections if it appears only twice. In general, the closer that f_1 is to 1, the better that a given term maps into a single category. Of course, aside from singular terms, few terms approach $f_1=1$. Moreover, if f_1 did equal 1 for a given term, that mapping would be of little value as a recall device (since any record containing that term could be obtained by searching on the section name). However, it still retains great value as a precision device, as it may still be used to partition records within the retrieved set. For example, suppose "estradiol" occurred only in the section on "Hormone Pharmacology". Then, all "estradiol" records could be retrieved by searching on the section name rather than on "estradiol". However, "estradiol" still separates records into two classes - with or without that term - and so it is still valuable for precision. In fact, using the data for Figure 20, the term "estradiol" peaks in Section 2, with 16 occurrences, and the second greatest peak occurs in Section 13, with 4 occurrences. The total number of occurrences of this term is 30. Hence, (except for the normalization),

$$f_1 = \frac{16}{30} = .533$$

$$f_2 = \frac{4}{30} = .133$$

$$f_1+f_2 = .666$$

So, for the term "estradiol", 66.6% of the unnormalized occurrences occurrences occur in two sections. Similar data for all terms is presented in Figure 25 through 34. For these calculations, the low frequency terms (less than 25 occurrences in two CA issues) were treated separately from the high frequency terms. The reason for this treatment is that low frequency terms may tend to occur in a small number of sections simply because they occur only a few times.

The high resolution of the term map suggests a method for overcoming the problem of selecting terms to feed back to the user that was identified in the first experiment. The searcher has only to name a term class of interest (e.g. "Hormone Pharmacology") and only the terms that belong to that class (such as "estradiol") and are present in the retrieved set will be identified and sorted for feedback. This procedure would simultaneously focus attention on the key terms, distinguish between content-specific and content-nonspecific terms, and simulate the general mechanism by which context is specified in discourse.

The average value of f_1 for high frequency terms, from Figure 25 is about .55, which means that the average high frequency term has 55% of its occurrences in one section. Figure 25 shows a similar plot for the second peaks of the high frequency terms. Since a second peak must necessarily contain less than half the occurrences of a given term, the curve falls to zero somewhat short of f_2=.50 (actually at f_2=.48). The average value of f_2 for high frequency terms is, from the data of Figure 26, is about .12 5, so that about 68% (f_1+f_2) of high frequency term occurrences are accounted for by the first and second peaks.

Figure 27 and 28 contain similar data for the low frequency terms. As expected, the very large component of low frequency terms that maps uniquely into a single section (f_1=1) is composed almost entirely (over 95%) of terms that occur only once. Most of the high frequency terms that map uniquely into

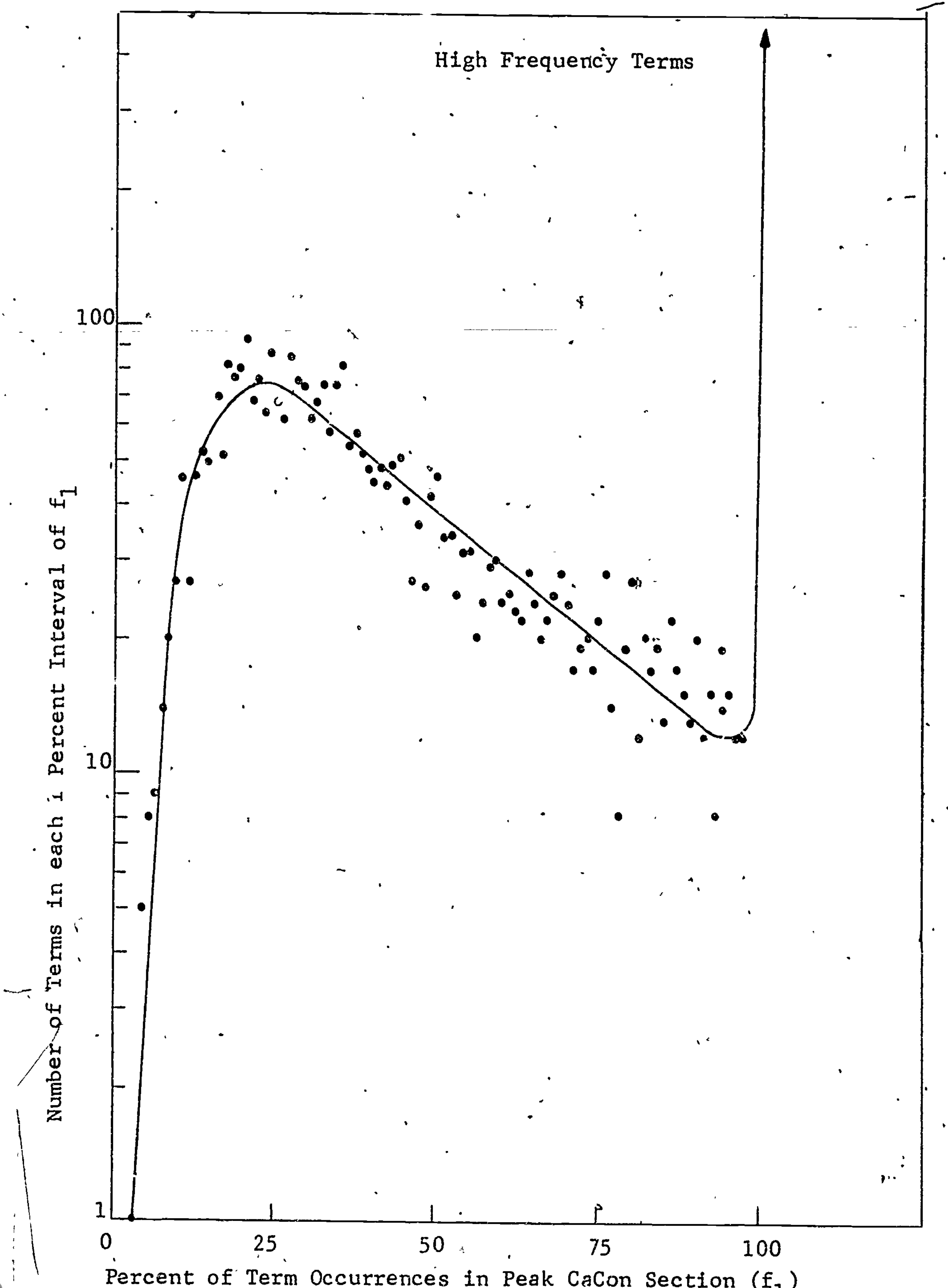


Figure 25. Distribution of All High-Frequency Term Largest Peaks in CACon Sections

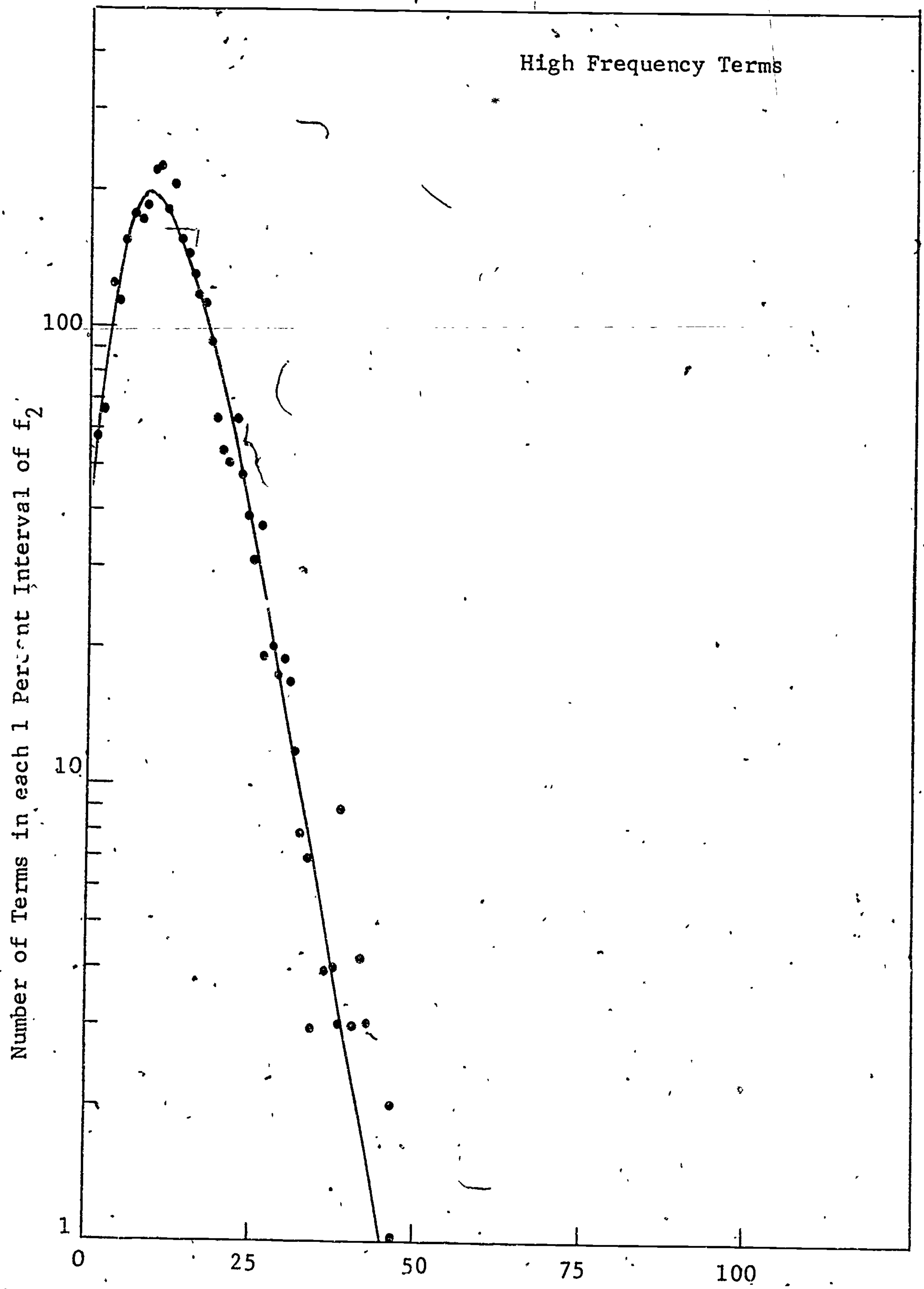


Figure 26. Distribution of all High Frequency Term Second Largest Peaks in CACon Sections

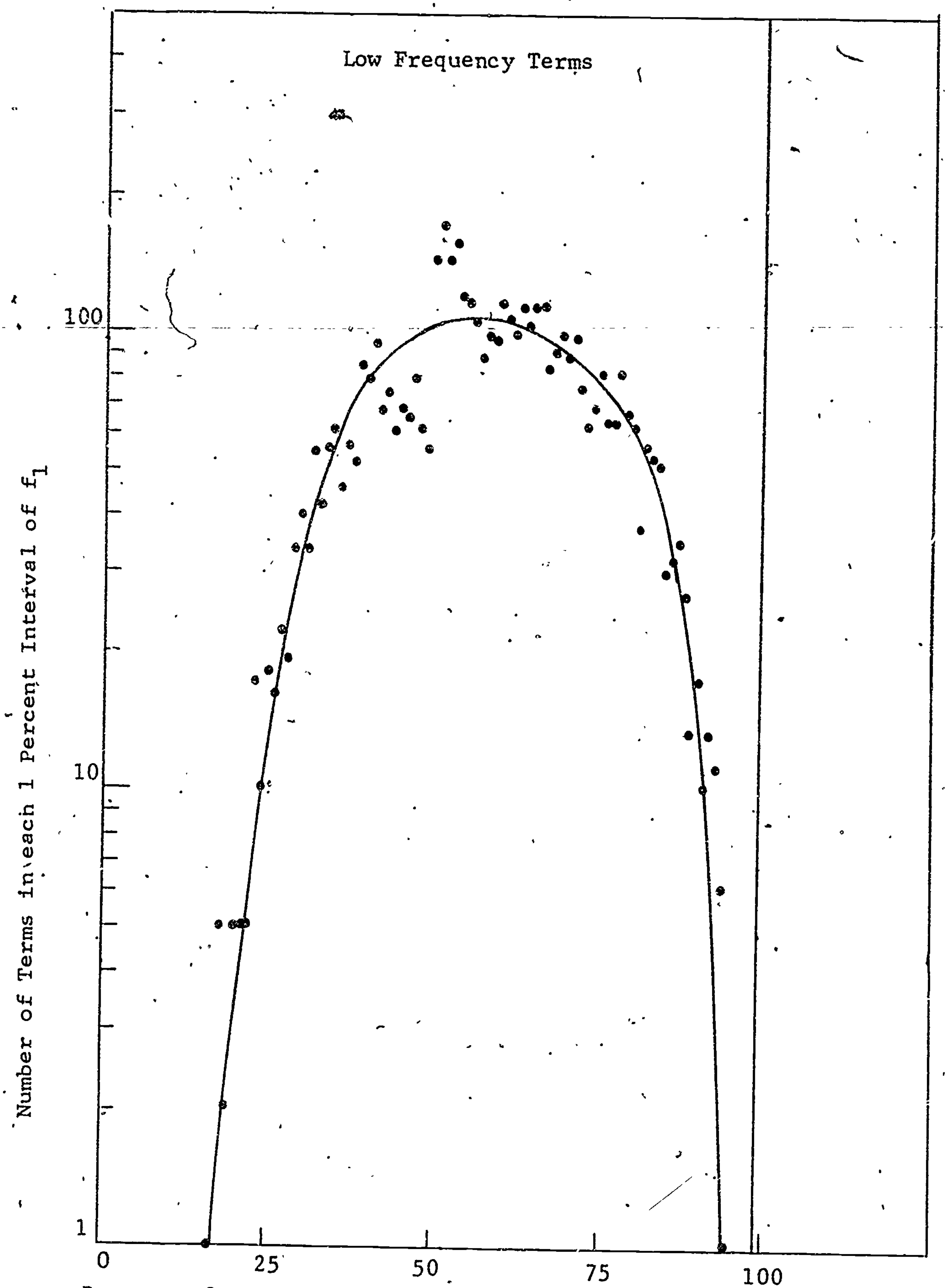
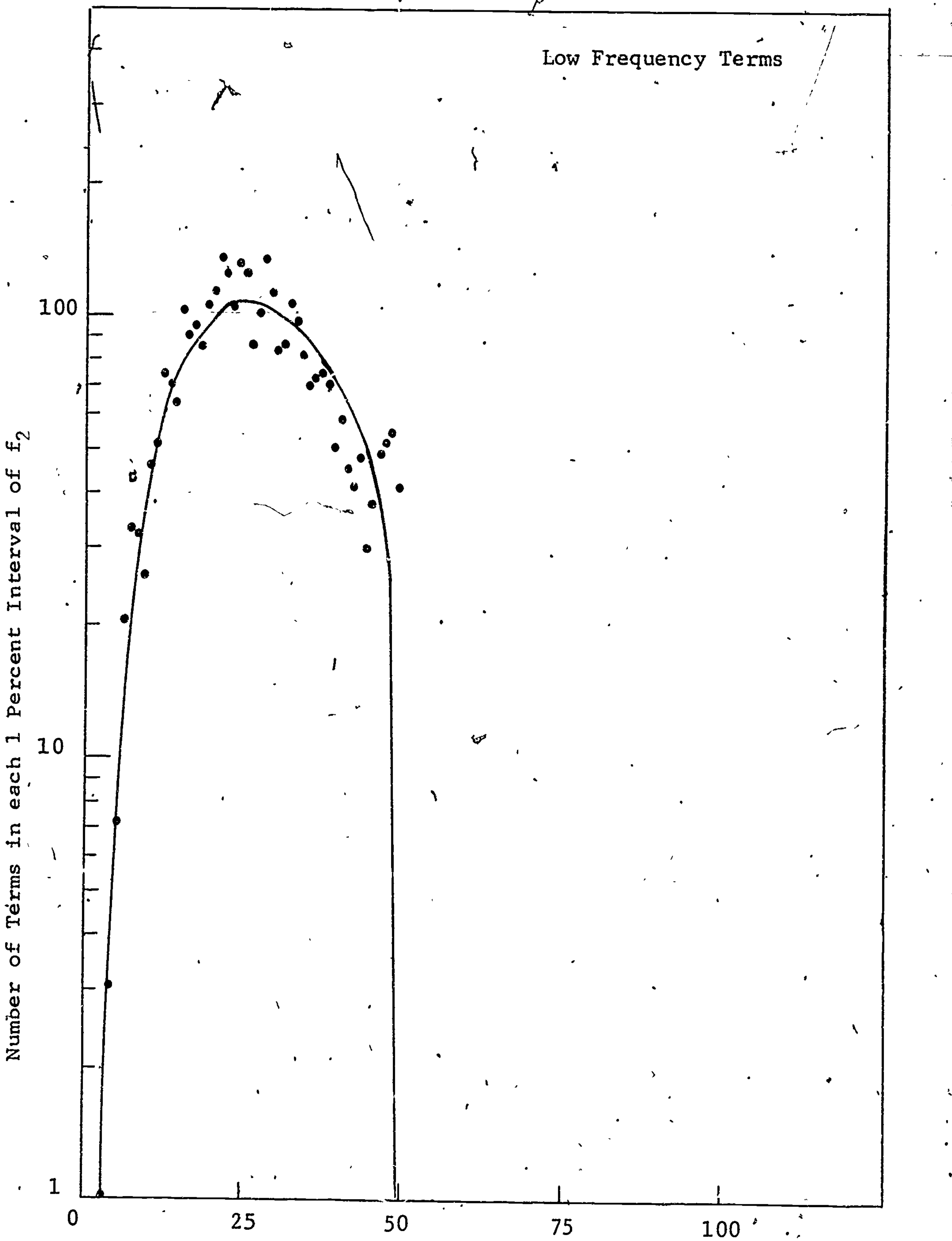


Figure 27. Distribution of all Low Frequency Terms Largest Peaks in CACon Sections



…e 28. Distribution of all Low Frequency Term Second Largest Peaks in CACon Sections one section are indexing terms that are assigned by CAS to the records.

Figure 29 presents data for the values of f_1 for the high frequency terms. The distribution is remarkably smooth and well behaved, and it shows that the concept of ATC is likely to work because so many terms have such large fractions of their occurrences in single sections. More than half of the high frequency terms each have more than half of their normalized occurrences in a single section. Since there are 80 total sections, the average fraction of term occurrences that would be expected in a section of the basis of chance for a randomized distribution of terms (no significant correlation of term occurrences) is only 0.013 (i.e. 1/80). In contrast to the observation that most term occurrences are uncorrelated with each other,[28,29] the correlation between terms and sections is very high.

Examination of the terms that have low values of f_1 reveals that they are the very general terms, such as "theory", "review", "experiment", "effect", etc. These terms should not map well, and the mapping technique provides a convenient method for isolating them. It is these high frequency terms which are not context specific that degrade the contribution of the abstract field to the resolution of records in Experiment 2. The mapping experiment (4) provides an easy method by which these terms could be grouped into a separate category from the context specific terms. If this were done, the resolution contribution of the abstract field should assume its expected dominant position among fields. Even discounting all the terms with $f_1=1$, the remaining low frequency terms average $f_1=61$ so that the low frequency terms (even excluding terms that occur only once) map very well into just one section each. Also, low frequency terms average $f_2=25$ so that, excluding terms that occur only once, about 86% of normalized low frequency term occurrences are in only two sections per term.

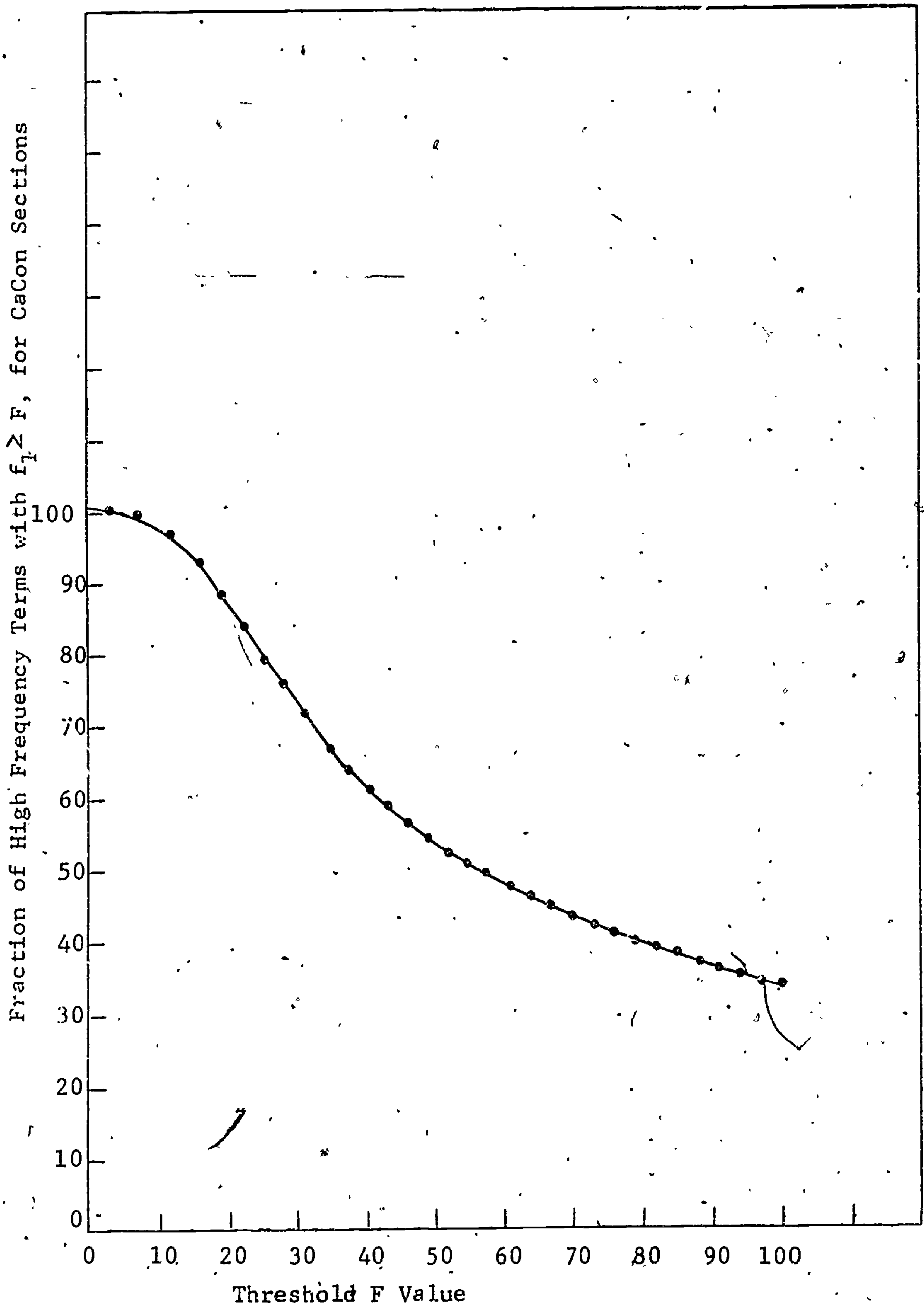


Figure 29. Fraction of High Frequency Terms With Largest Peaks Greater Than a Threshold

Figures 29 and 30 present the cumulative frequencies for the high and low frequency terms. That is, suppose that a threshold were set (F_1), and only terms with $f_1 > F_1$ were mapped. How many terms would be mapped for a given F_1? Figures 29 and 30 give the answer. For instance, if $F_1=0.3$, then 72% of the high frequency terms and virtually all the low frequency terms would be mapped.

Note that this result is in harmony with the intuitive notion that the lower frequency terms are more content specific, for the occurrences of the average low frequency term are more concentrated into a single section than are the occurrences of the average high frequency term.

Figures 31 through 34 contain similar data for the distribution of terms over supersections. Since each supersection is composed of several sections, the fraction of occurrences in a given division, f_1, must be greater or equal for supersections as opposed to sections. Remarkably, 94.7% of high frequency terms map into one supersection with $f_1 > .99$. A similar statement also holds true for the low frequency terms, distributed over supersections. Clearly, the supersection division of terms is much less demanding than the section division and denotes a second very valuable level to the mapping hierarchy.

The vocabulary mapping experiments show that simple statistical sorting operations applied to manually indexed data base can yield a very useful hierarchical mapping of the terms into categories. It now remains to be shown that these categories prove useful for the IR tasks that have motivated their construction. In the spirit of the previous discussion, the statistical intellectual term classes offer the following method for overcoming the limitations of binary comparison. For the example of "dog", "greyhound" and "bean", the first two terms map into the "Mammalian Biochemistry" sections of CACon (CA011). "Bean" maps into the "Plant Biochemistry" section of CACon (CA017). As before, "maps" means that the

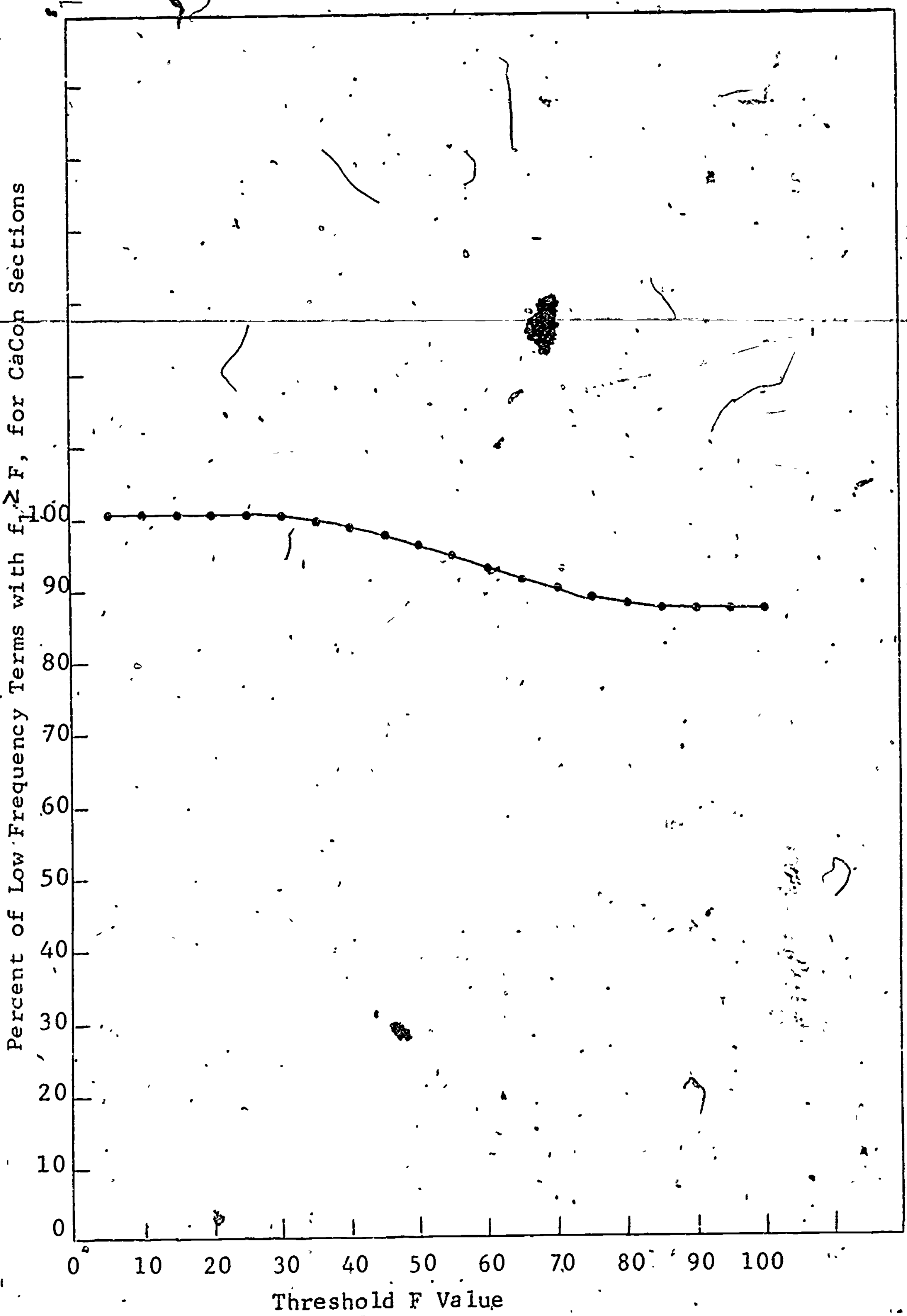


Figure 20. Fraction of Low Frequency Terms With Largest Peaks Greater Than A Threshold

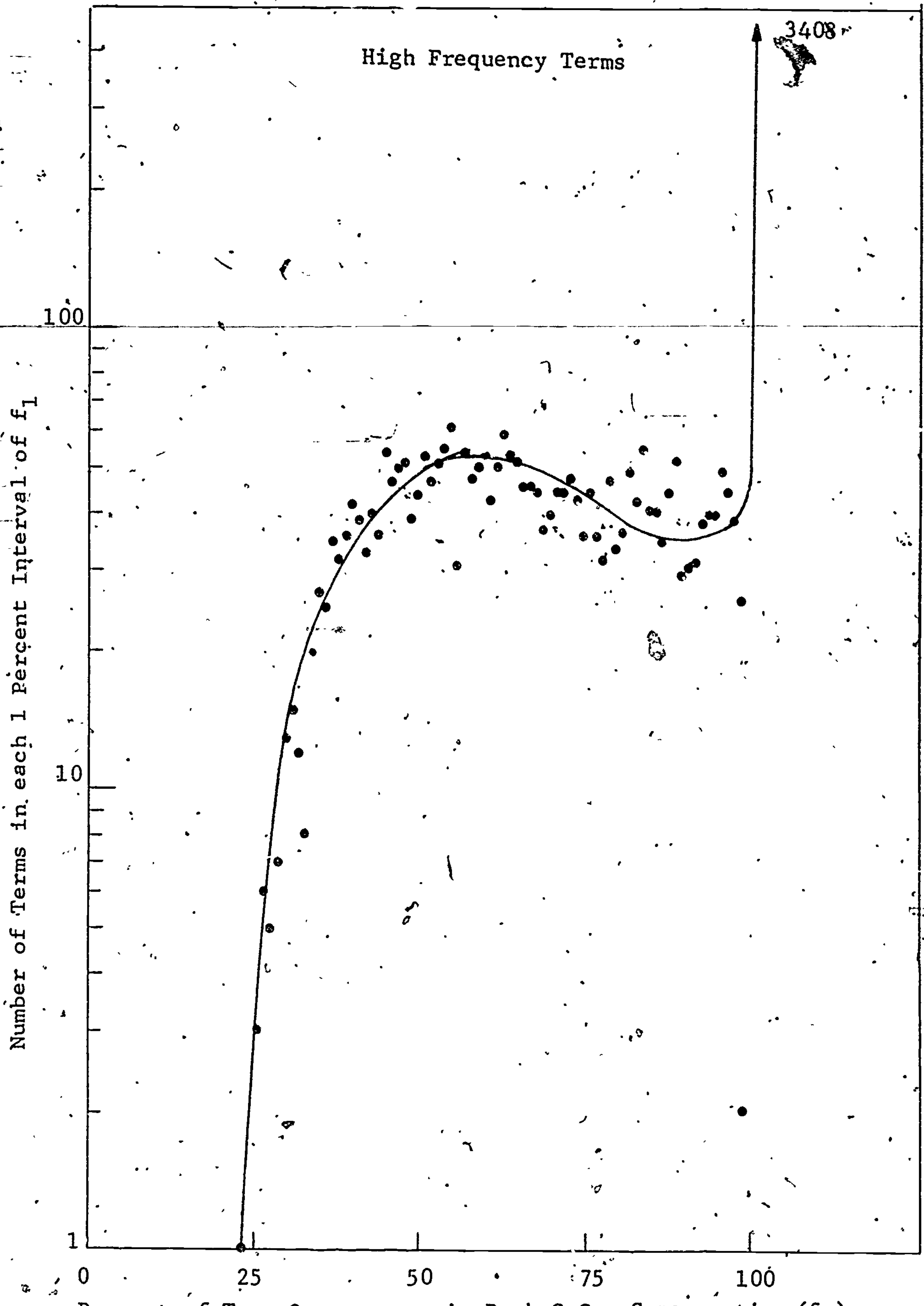


Figure 31. Distribution of All High Frequency Terms Largest Peaks in CACon Supersections

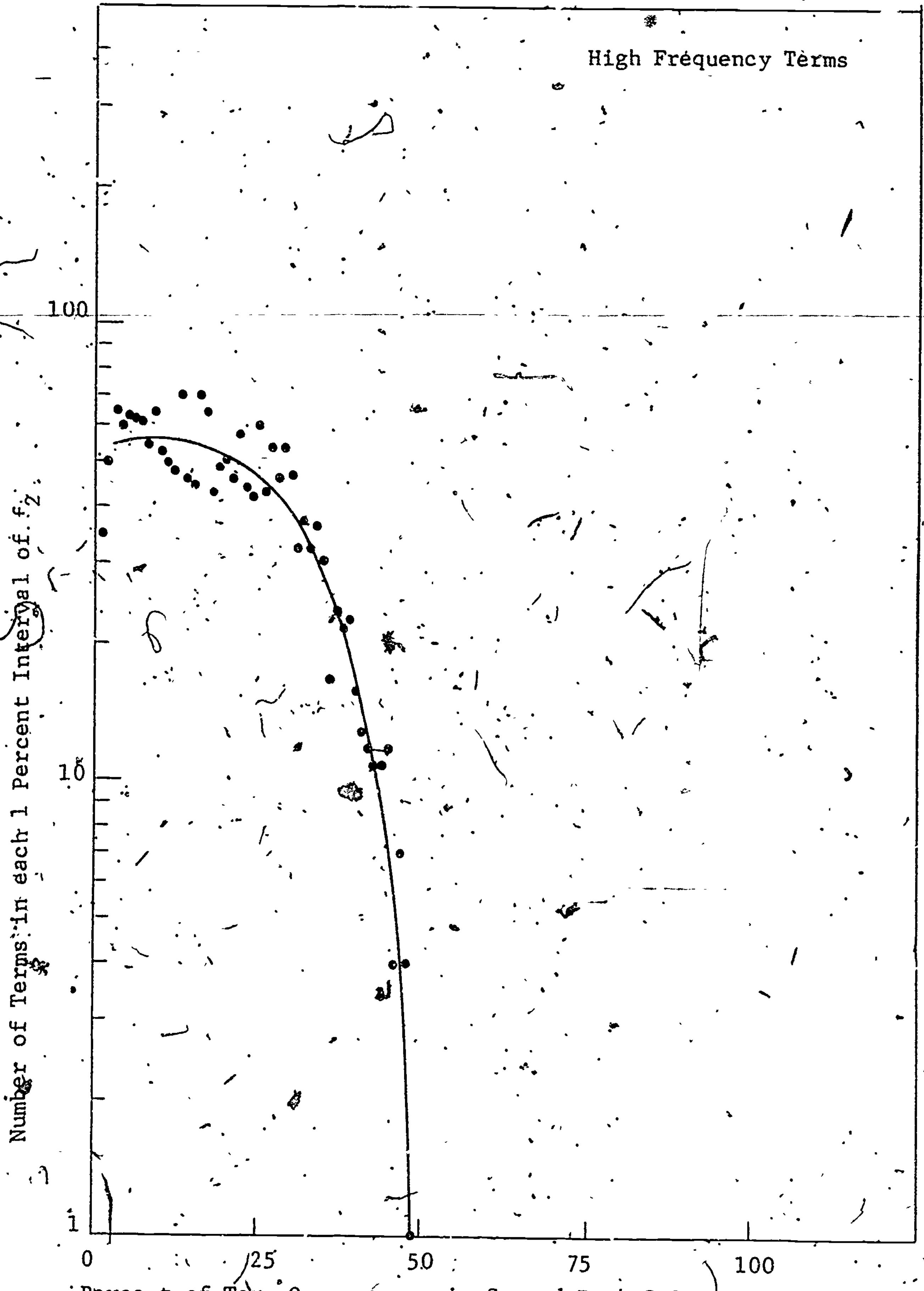


Figure 32. Distribution of All High Frequency Terms Second Largest Peaks in CACon Supersections

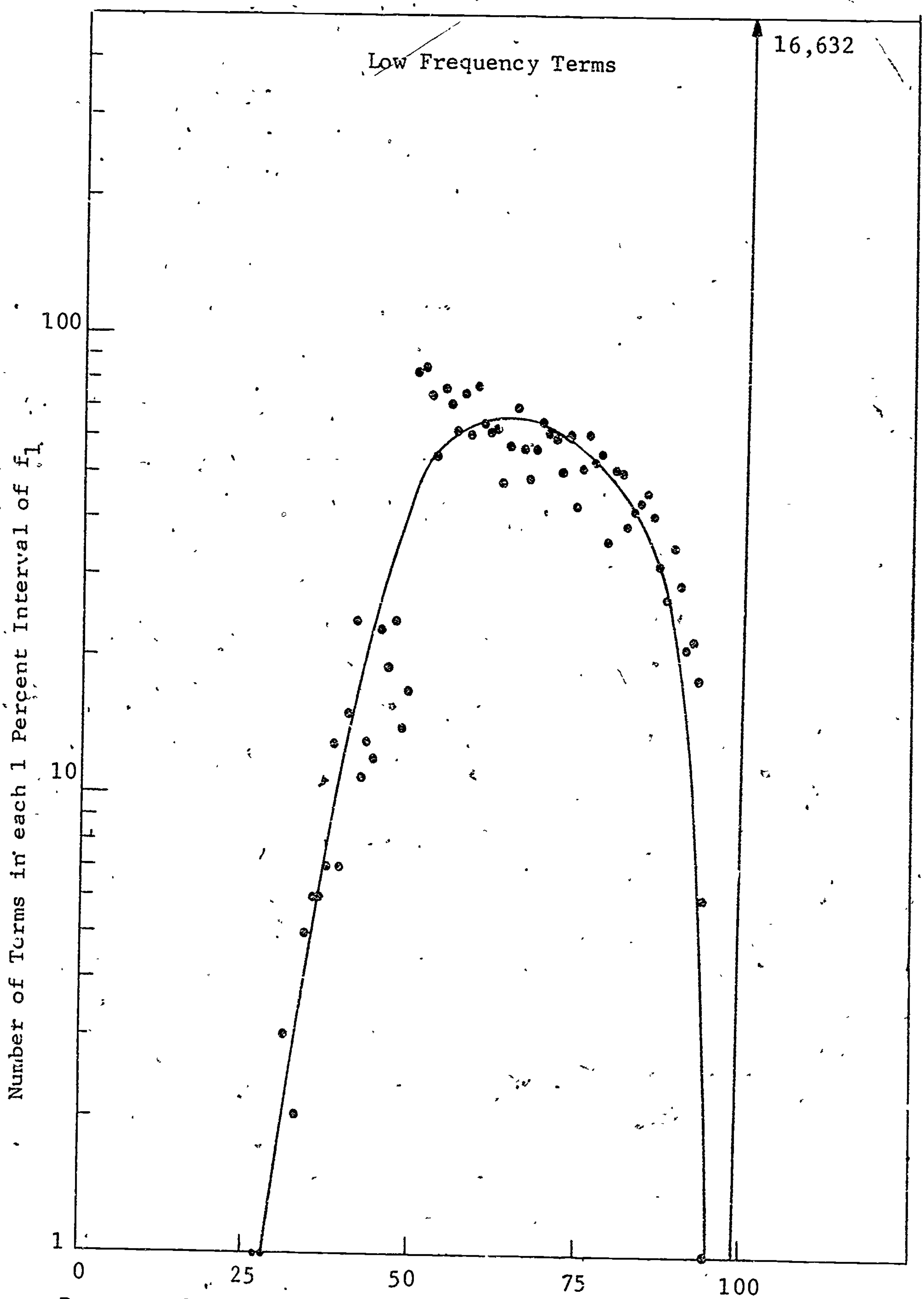


Figure 33. Distribution of All Low Frequency Terms Largest Peaks in CACon Supersections.

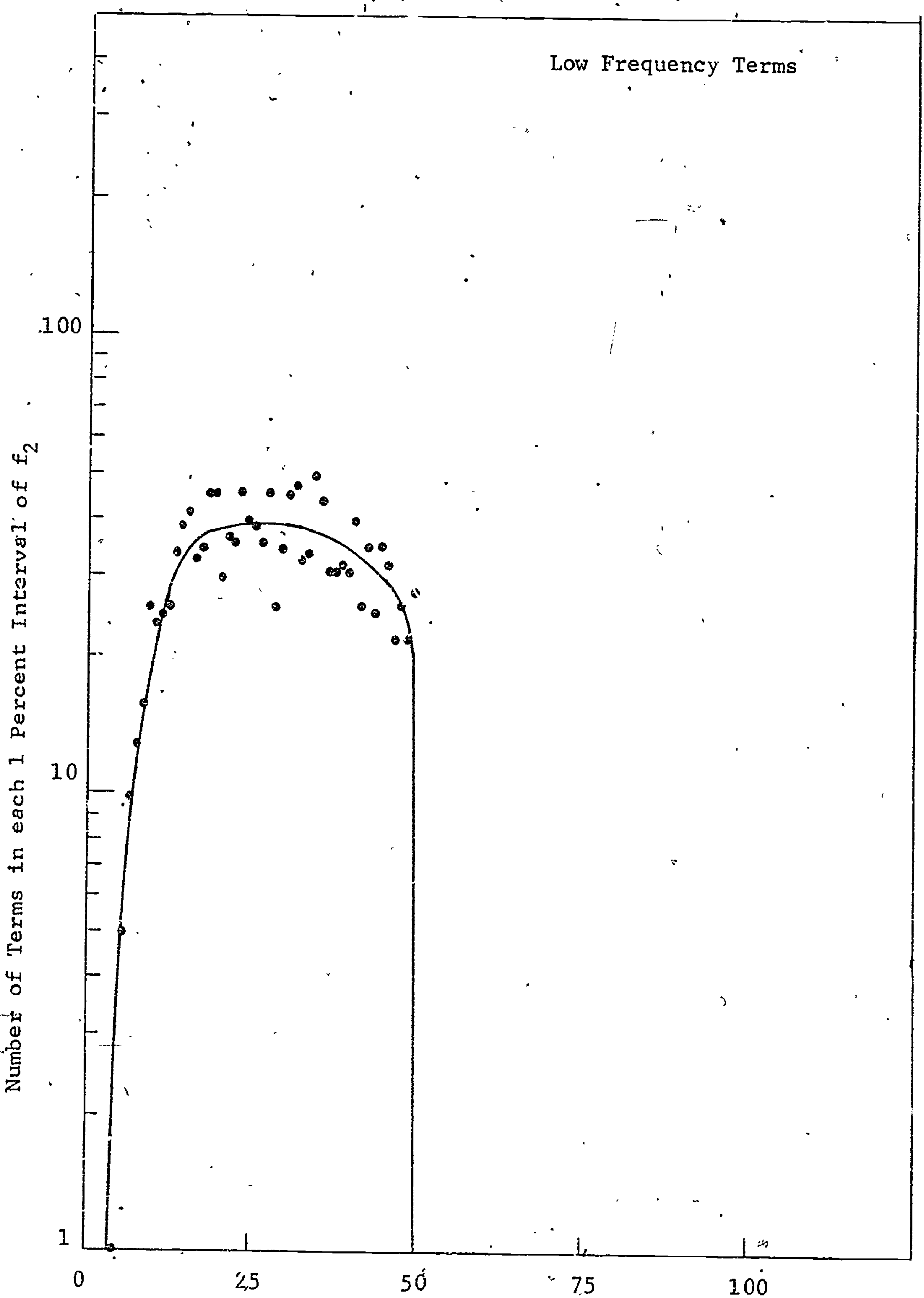


Fig. 34. Distribution of All Low Frequency Terms Second Largest Peaks in CACon Supersections

term has its greatest concentration in the given section. Now, if each term is augmented by adding the class name to it, the following situation arises:

Bean	Bean-CA017
Dog	Dog-CA011
Greyhound	Greyhound-CA011
No matches	One link between Dog and Greyhound at distance = $1 - \frac{1}{3} = 0.67$

That is, "dog" is linked to "greyhound" at a distance intermediate between identical match and no match. Augmented identical terms still match at zero distance.

The principle of augmented terms can be applied at more than one level. Thus, a term can be autmented with the names, for instance, of the CACon subsection, section and supersection in which it occurs so:

Term 1 · CACon Subsection 1 · CACon Section 1 · CACon Supersection 1
Term 2 · CACon Subsection 2 · CACon Section 2 · CACon Supersection 2

If Term 1 is identical to Term 2, they are joined at distance zero. If Term 1 is not equal to Term 2, but they map into the same subsection (So CASub 1 = CASub 2. CASect 1 = CASect 2 and CASuper 1 = CASuper 2) then Term 1 and Term 2 are joined at distance = $1 - \frac{3}{5} = .4$. Similarly, if the CASuper's are equal, the connection is at distance = $1 - \frac{1}{7} = 0.86$. The progressive distances of the connections joins at different levels of map relatedness are in close correspondence with intuitive expectations of desired term behavior. Moreover, the simplicity of the procedures means that they can be performed inexpensively.

4. ANALYSIS

The three critical parameters that characterize a clustering run are coverage, agglomeration and accuracy. By using a statistical model of the clustering process (assuming that term occurrences are largely uncorrelated), and a simple measure of term distribution, it is possible to predict the coverage and the agglomeration as a function of the cluster distance. The model also predicts which terms will be dominant in forming the pattern and leads to recommendations for modification of the shape of the term frequency distribution to improve retrieval efficiency. The model does not predict the accuracy of record assignment to clusters. However, one can readily use the model to calculate the degree by which an experimentally determined set of assignments exceeds the chance level. By using experimentally determined clustering accuracy as a function of measures of the term distribution, estimates of the usefulness of clustering in new situations can be made. The excellence of the agreement between the model and the data supports the assumption of uncorrelated term occurrences, in support of the literature[28,29].

STATISTICAL MODEL OF CLUSTERING COVERAGE

1. All Term Frequencies Equal

Suppose that in a collection of N_F records, there are J unique terms, each of which occurs with the same frequency, N_j (i.e. each of the J terms occurs in the same number of records). The case of equifrequent terms is simple to test, and can readily be generalized to describe the case wherein the terms each have their own frequencies (each term may occur in a different number of records). Moreover, assume that each record has the same number of terms, $\bar{N}_T$. This is a good assumption for the CACon data base. Note that $\bar{N}_T = \frac{N_j \cdot J}{N_F}$.

Represent each record by a J-tuple. Let a 1 in the jth position correspond to the presence of the jth term, and let a 0 correspond to its absence. For each record, the corresponding J-tuple will have $\bar{N}_T$ of its positions filled with 1's. To calculate the number of records that are clustered at a given distance, one merely has to calculate the number of records that share at least k terms with at least 1 other record, where k is determined by the distance formula

$$D = 1 - \frac{k}{2\bar{N}_T - k}$$

$$\text{So } k = 2\bar{N}_T(1-D)/(2-D)$$

Given any two records from the collection, the probability that they will match on at least one term is easily calculated. Since all the terms have equal frequencies, the probability that any one term is present in a given record is the same problem as the probability of picking one specified ball in $\bar{N}_T$ chances from an urn with J numbered balls.

The probability that there is a match on the jth term is the product of the probabilities that the jth term is present in each of the two records. Let:

$p(j)$ = probability of a match on the jth term

$p(j)$ = (probability that the jth term is in R_1) · (Probability the jth term is in R_2 given that it is in R_1)

$$p(j) = \frac{N_j}{N_F} \cdot \frac{N_j - 1}{N_F - 1}$$

$\underline{P}(k)$ = probability that there are at least k term matches between R_1 and R_2

$\underline{P}(ex\ k)$ = probability that there are exactly k term matches between R_1 and R_2

$\underline{P}(ex\ 0) = 1 - ((1-p(j))^J$

That is the probability that there are no term matches between two records is 1 minus the product of the probabilities that there is no term match on any of the J terms.

$$Ln\left[1-\underline{P}(ex\ 0)\right] = Ln\left[((1-p(j))^J\right] = J\,ln(1-p(j))$$

for $p(j) \ll 1$, $Ln(1-p(j)) \simeq -p(j)$

So $Ln\left[1-\underline{P}(ex\ 0)\right] \simeq -Jp(j)$

$1 - \underline{P}(ex\ 0) \simeq exp(-Jp(j))$

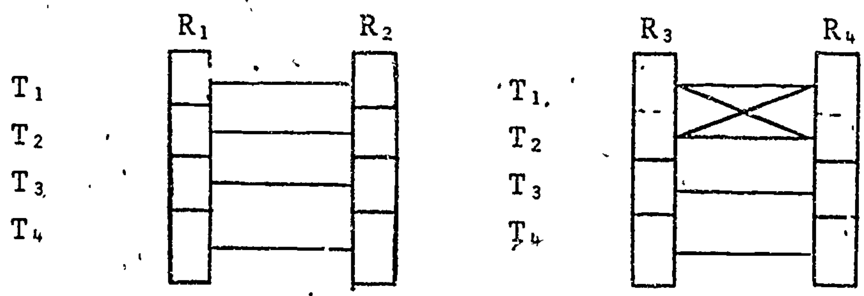
$\underline{P}(ex\ 0) = 1 - exp(-Jp(j))$

So, the probability of at least one match is 1 minus the probability of no matches, and

$\underline{P}(1) = exp(-Jp(j))$

$$\underline{P}(1) = exp\left[-J \cdot \frac{N_j}{N_F} \cdot \frac{N_j - 1}{N_F - 1}\right]$$

Case of Non-Equal Term Frequencies



As an example of partial record sets with terms of unequal frequency, consider records pairs ($R_1 + R_2$) and ($R_3 + R_4$): For (R_1 and R_2) there are 4 possible terms ($J = 4$), all of equal frequency. Suppose $\bar{N}_T = 1$. Then there are 4 matches out of 16 possible combinations for a match probability of $\frac{4}{16} = \frac{1}{4}$ at a distance of $1 - \frac{k}{2\bar{N}_T - 1} = 1 - \frac{1}{2-1} = .0$. Suppose that R_3 and R_4 are identical to R_1 and R_2, except that the first two terms are identical, (i.e. the first term has twice the frequency of any of the others). Thus, there are, in effect, 3 terms ($j = 3$), one of which has twice the frequency of the other two. From the diagram, there are 6 matches out of 16 possible combinations for a match probability of $\frac{6}{16} = \frac{3}{8}$. So, it is clear that for cases of unequal frequency, each term contributes to the matches approximately according to the square of the term frequency.

When the derivation of P_k' is done for the case where the terms are each allowed to have distinct frequencies, (See Appendix B) the result is found to obey a Poisson distribution.

$$P(k) = 1 - \sum_{k=0}^{K-1} \frac{\bar{L}^{k} e^{-\bar{L}}}{k!} \quad \text{for } k>1 \text{ and } \frac{N_j}{N_F} \ll 1 \text{ for all } j$$

for N_j comparable to N_F, (which corresponds to the case where one term occurs in most records), additional factors of $\bar{L}$

occur in the result. In this expression,

$\bar{L}$ = the average number of term matches (links) per record pair.

Since the number of record pairs is $\frac{N_F(N_F-1)}{2}$ and the number of term matches is $\frac{\sum_{j=1}^{J} N_j(N_j-1)}{2}$, $\bar{L} = \frac{\sum_{j=1}^{J} N_j(N_j-1)}{N_F(N_F-1)}$

It is useful to note that the equation for $P(k)$ depends only on the parameter $\bar{L}$. Since the shape of the cluster pattern depends on the number of links formed, one may ask which terms contribute most to the formation of a pattern. Clearly the single frequency (one appearance only) terms cannot contribute much to a pattern since they cannot produce a link. It has been argued by others that such terms contribute to the pattern by identifying dimensions along which records are different[19]. That is true, but the experiments show that terms are so weakly semantically linked that singular terms only degrade the pattern, i.e. degrade the significance of the matches.

Higher frequency terms contribute progressively more to a pattern. A term with a record frequency of N_j contributes a number of links

$$L = \begin{bmatrix} N_j \\ 2 \end{bmatrix} = \frac{N_j(N_j-1)}{2}$$

For $N_j = 1$ (singluar terms) it is zero. For $N_j >> 1$, as expected, it increases as N_j^2. Because there are very many more low frequency terms control the overall cluster pattern for a given case. Figures 37 and 38 indicate that sometimes even a single high frequency term can overbalance the lin g power of all the low frequency terms. This work suggest hat it is not sufficient to report $\bar{N}_T$, N_F, J and $\bar{N}_j$ when documenting clustering experiments. It is also desirable to report the average number of links per record pair, $(\overline{L})$

If there are any terms in the file for which $N_R \simeq N_F$, these should be reported too (See Appendix B). It is for this reason that typical distributions, rather than average distributions are plotted in Figures 35 and 36, i.e. since

$\overline{N_j^2} \gg \overline{N}_j^2$, to calculate $\overline{L}$ on the basis of average term frequencies would underestimate the significance of the high frequency terms.

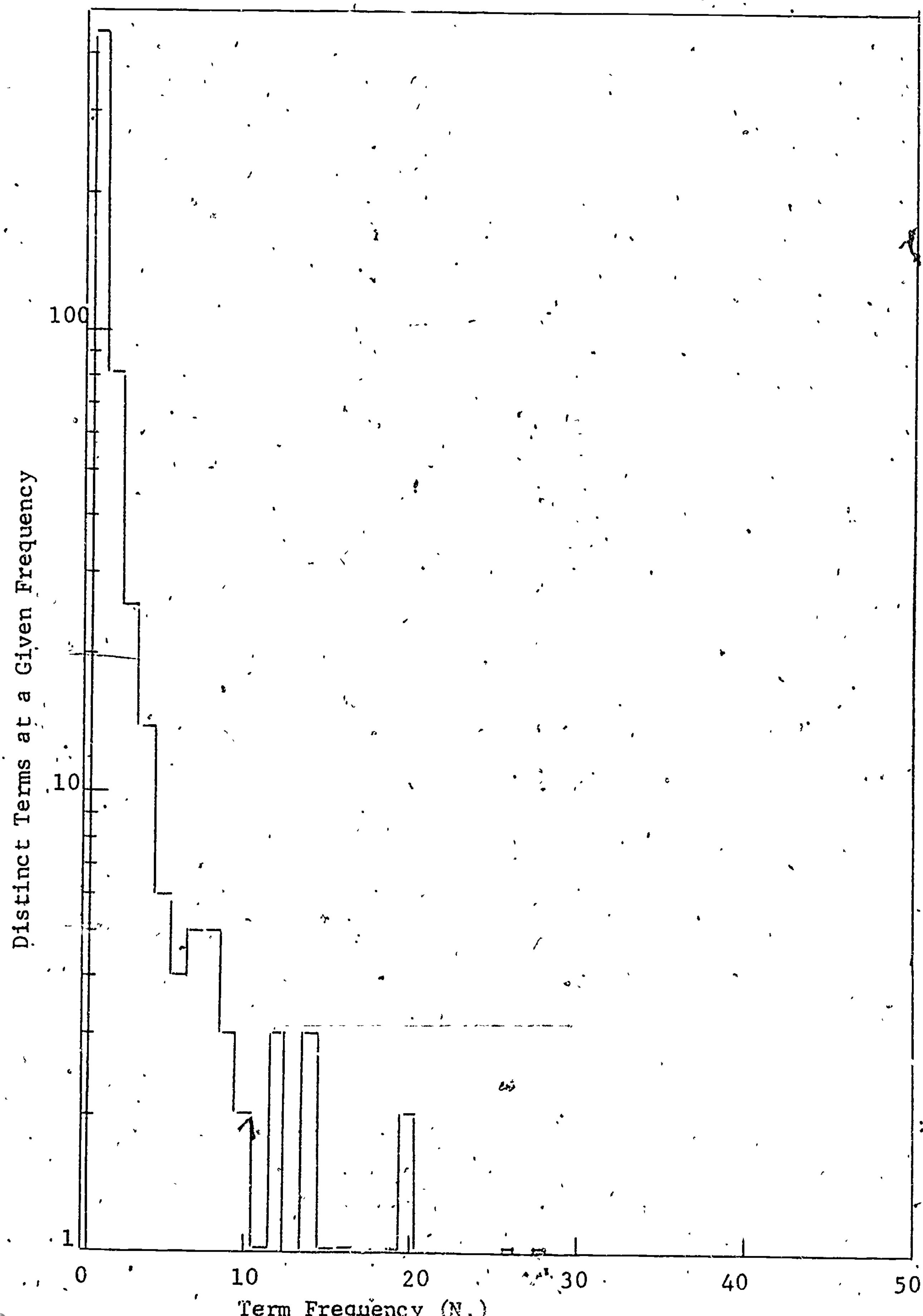


Figure 35 A Typical Distribution of Term Frequency for 100 CACon Records

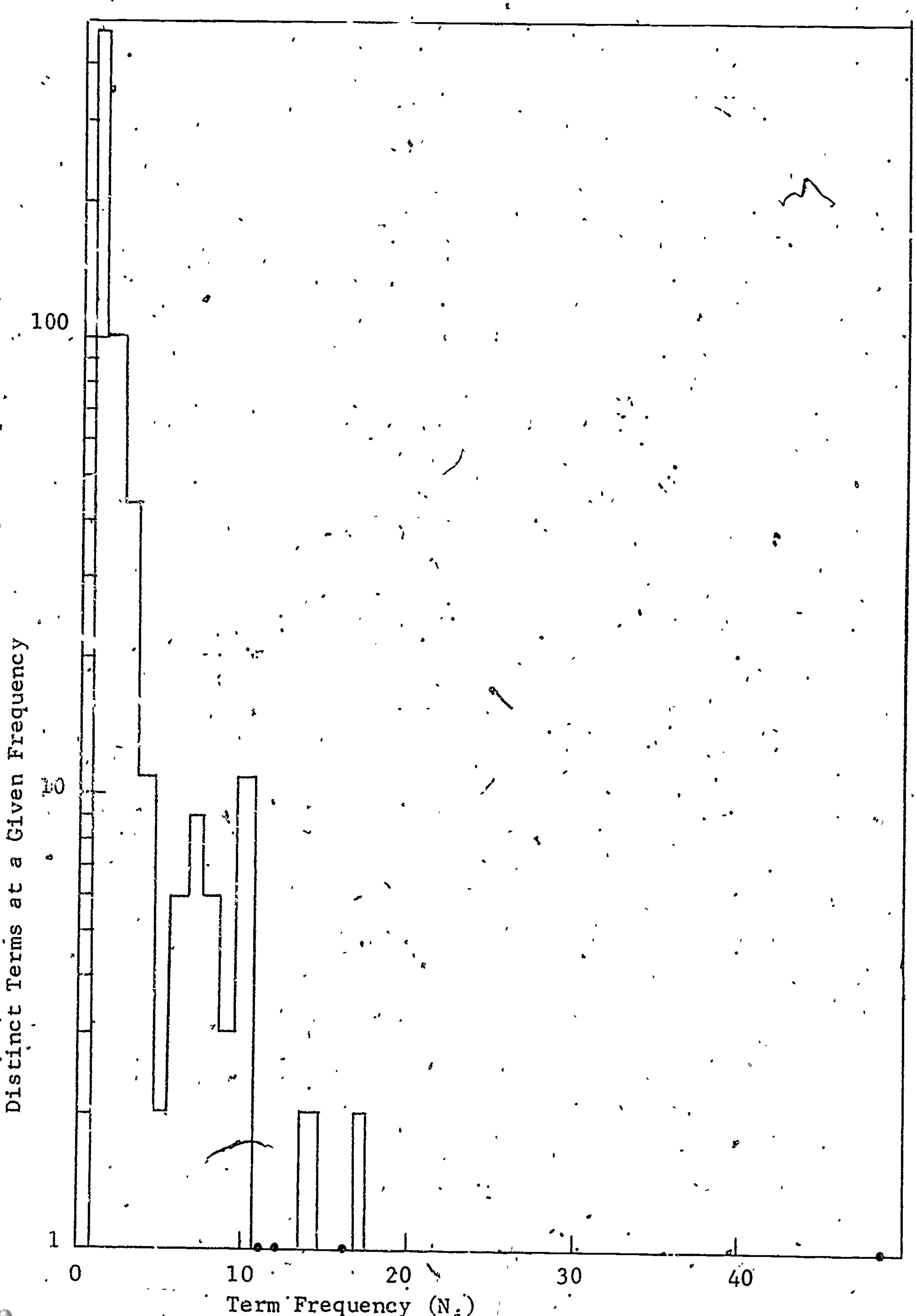
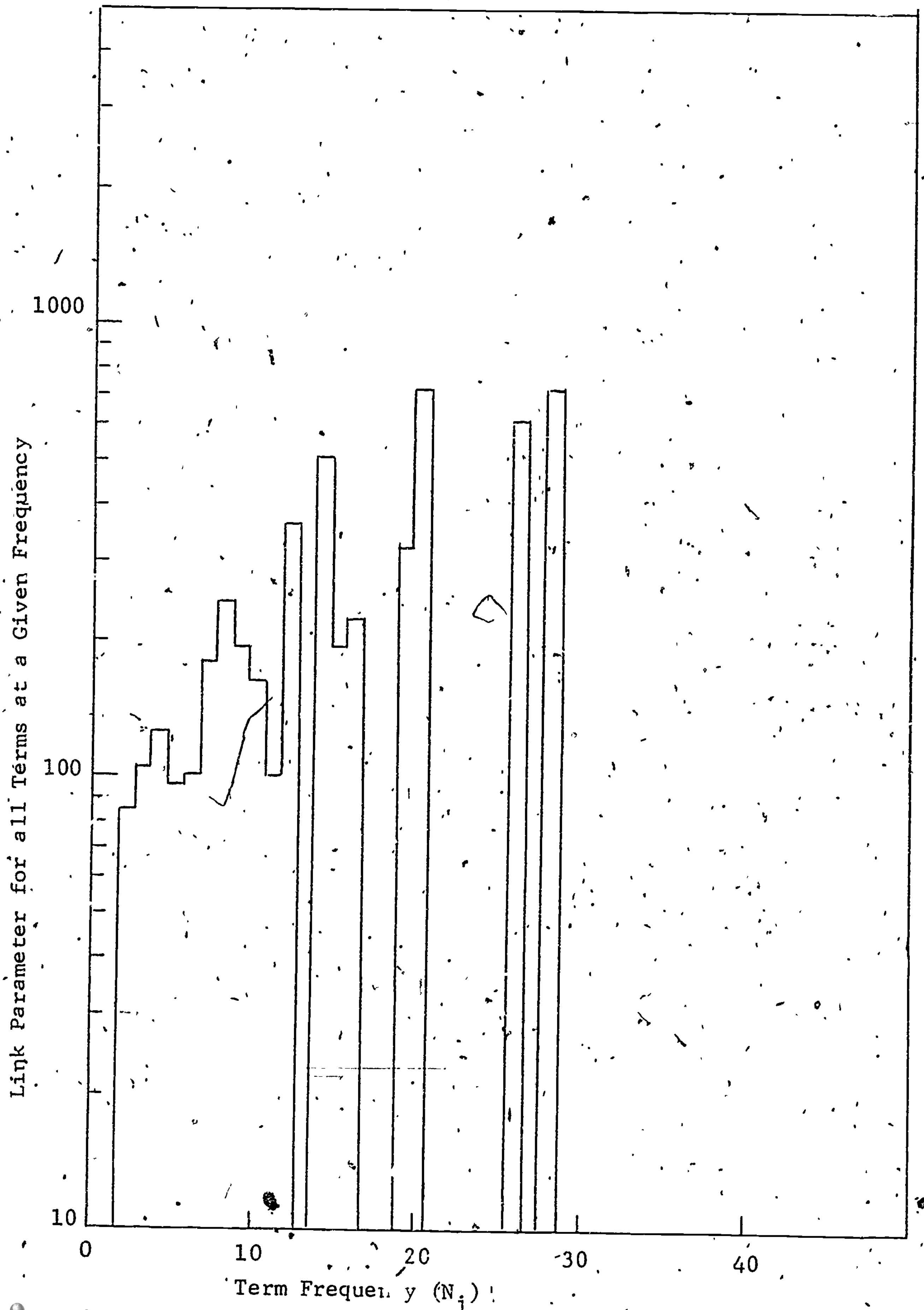


Figure 36. A Typical Distribution of Term Frequency for 100 CACon Records



re 37. A Typical Distribution of Term Linking Power for 100 CACon Records
.1

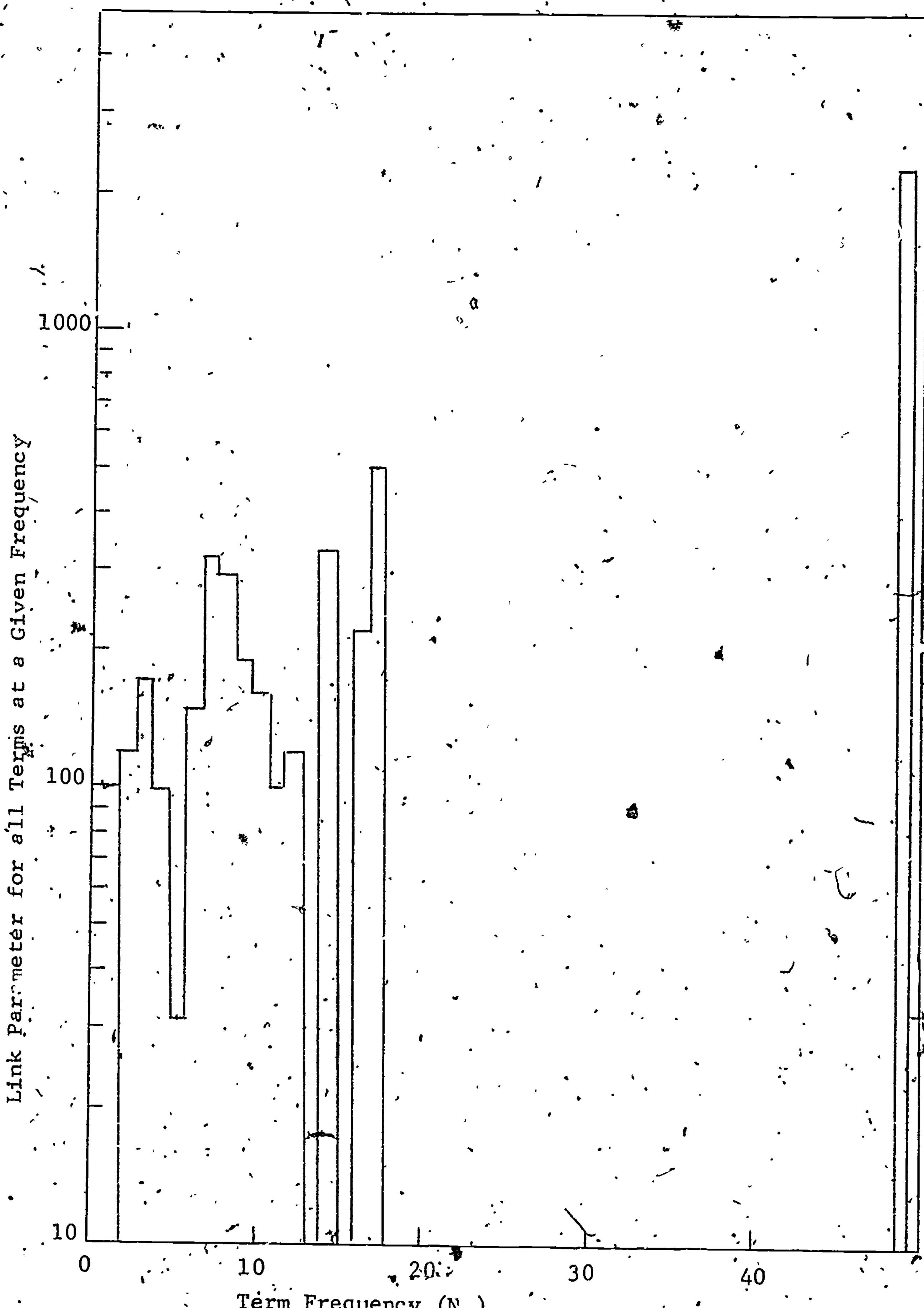


Figure 38. A Typical Distribution of Term Linking Power for 100 CACon Records

Number of Records Clustered - Multiple Links and Agglomeration

Now, the number of records clustered at a given distance D_1, N_c = (Prob of at least k links between R_i and R_j) · (Number of R_i and R_j pairs) · (Number of records clustered per link) where k links assure $D \leq D_1$

$$N_{c_1} = P(R_1, R_2) \cdot N(R_1, R_2) \cdot f(L, N_F)$$

$f(L,M)$ expresses the fact that when new links are formed they may either involve previous linked records or not, as shown on Figure 39. Figure 40 expresses $f(L,M)$, calculated explicitly for $M=100$. Note that for $L \approx 0$, $\frac{\Delta N_c}{\Delta L} \sim 2$ because every new link is a type 1 link and binds two previously unbound records.

For $\frac{N_c}{N_F} \approx .6$, $\frac{\Delta N_c}{\Delta L} \sim 1$ because most new links are type 2 links, which bind one previously unbound record to other previously bound records.

For $\frac{N_c}{N_F} \sim 1$, $\frac{\Delta N_c}{\Delta L} \sim 0$

because new links occur primarily as type 3, which only bind previously bound groups together.

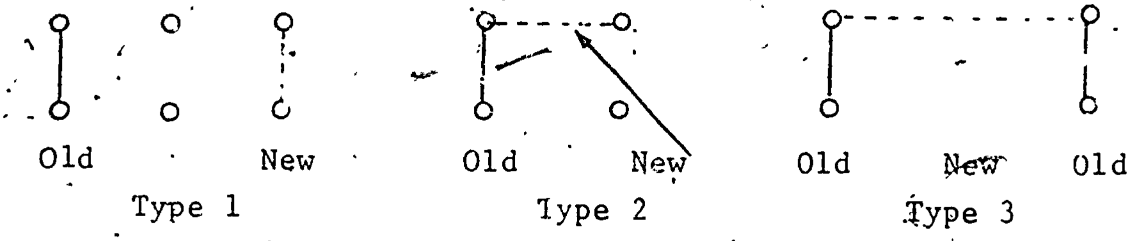


Figure 39. Types of Ways that New Links Can Occur

It has be shown by derivation and explicit calculation that it is roughly true that

$$N_c \sim N_F \left(1 - \exp\left(-\frac{2L}{N_F}\right)\right)$$

for $\frac{2L}{N_F} << 1$, $N_c \sim N_F \left(1 - \left(1 - \frac{2L}{N_F}\right)\right) \sim 2L$

for $\frac{2L}{N_F} >> 1$, $N_c \sim N_F$

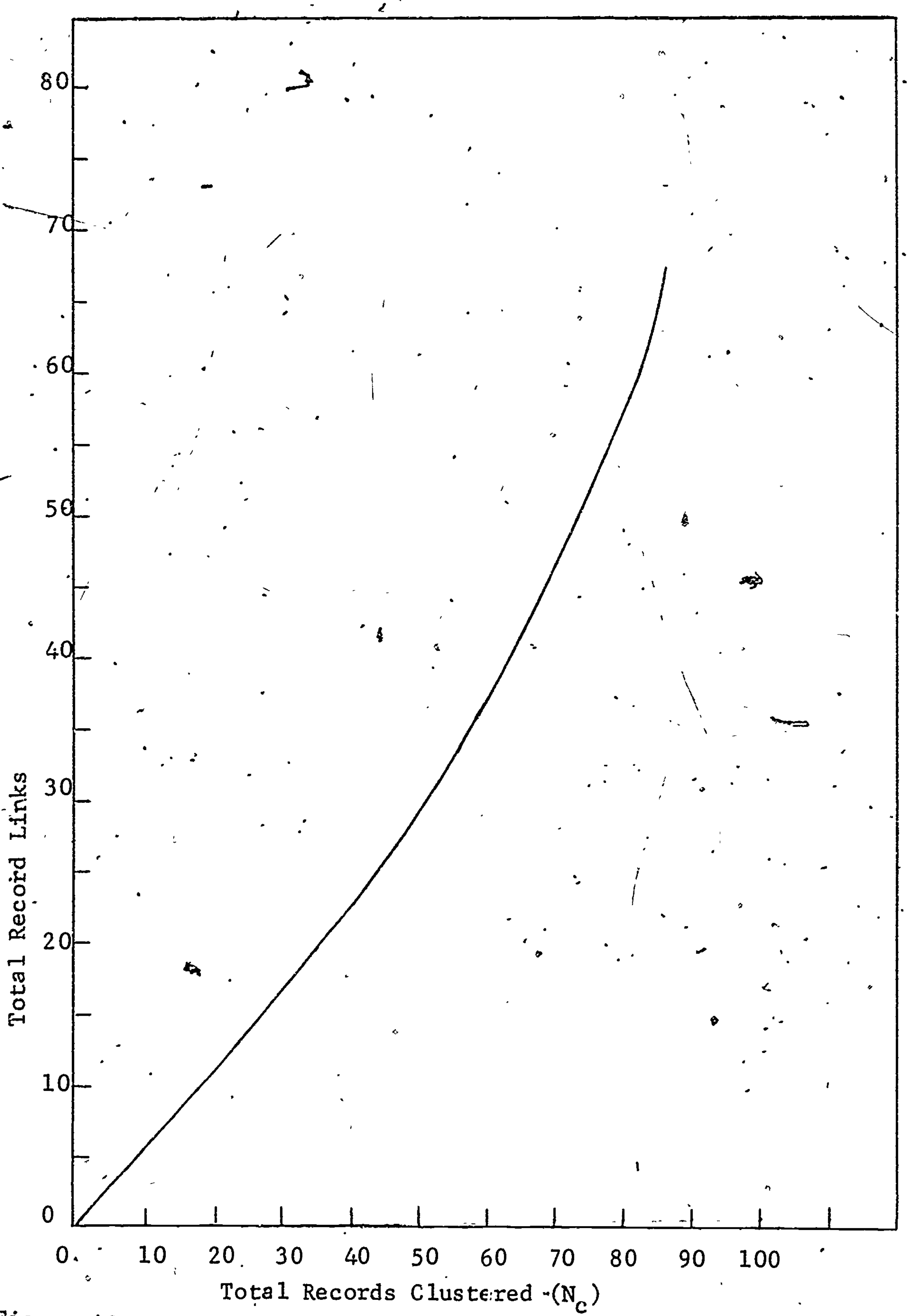


Figure 40. Link Redundancy Factor vs Number of Records Clustered for a 100 Record File

combining expressions,

$$N_c = \left\{1 - \sum_{k=0}^{k-1} \frac{L^k e^{-\bar{L}}}{k!}\right\} \cdot \frac{N_F(N_F-1)}{2} \cdot \frac{N_F}{L}\left\{1-\exp\cdot\frac{2L}{N_F}\right\}$$

The following graph shows the data of Figure 14. The line is that calculated using the above values. The curve matches the average of the relevant/nonrelevant experimental coverage within one standard deviation of the mean.

The coverage model was also tested on the data of Experiment 2. As shown in Figure 42, it fits the data well for various conditions.

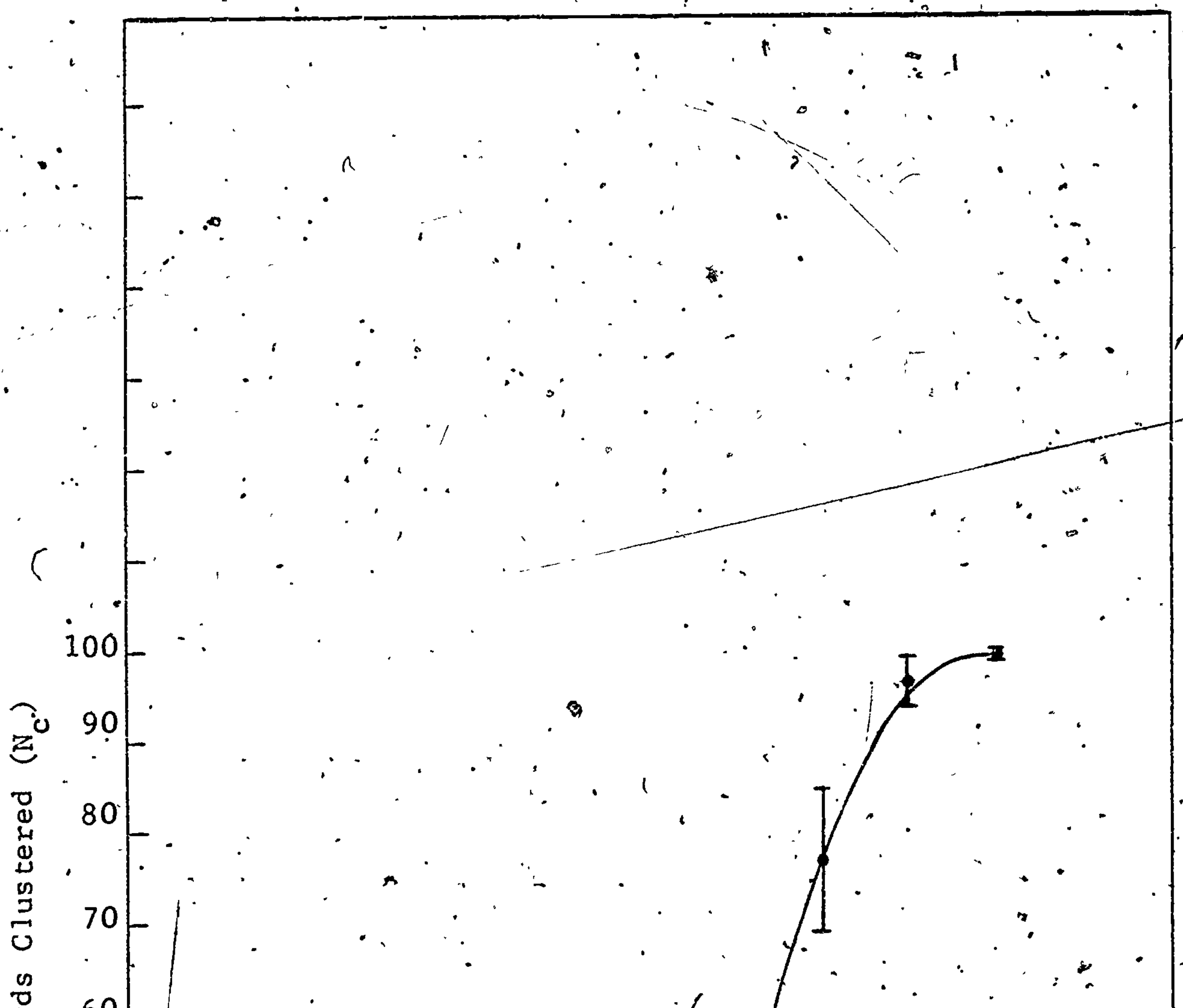


Figure 41. Fit of Statistical Coverage Model to Data .1

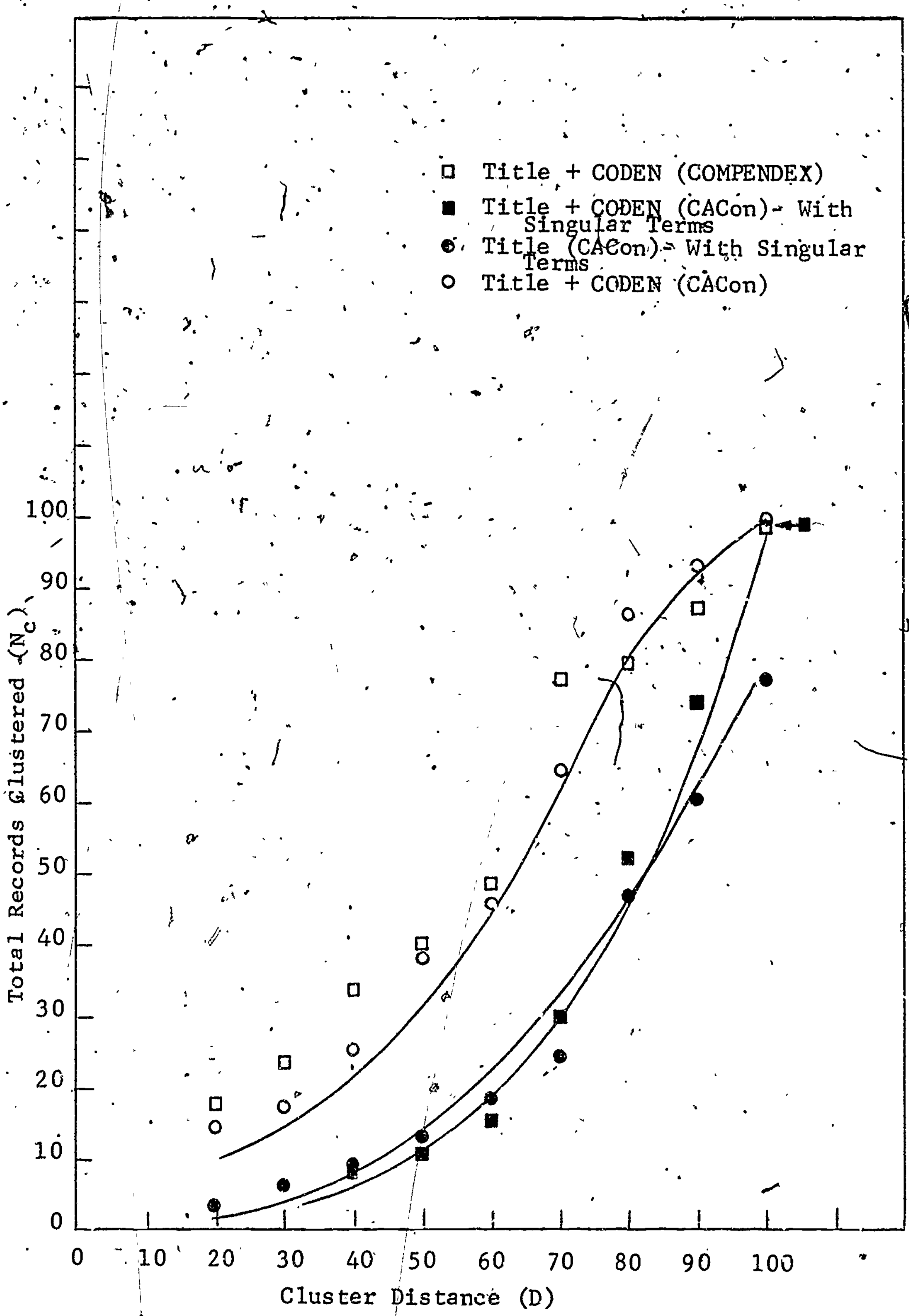


Figure 42. Fit of Statistical Coverage Model to Data .2

STATISTICAL MODEL OF AGGLOMERATION

The average cluster size depends on the number of type 1, type 2 and type 3 joins (n_1, n_2, n_3) respectively. (See Figure 35). The number of separate clusters is approximately $n_1 - n_3$, since an n_1 join creates a cluster and for $N_c << N$, an n_3 type join usually destroys one. An n_2 join neither creates nor destroys a separate cluster, but rather it just joins a previously unjoined record to an existant cluster. Hence, the average number of records per cluster, N_A, is given approximately by:

$$N_A \cong \frac{N_c}{n_1 - n_3} \quad \text{for } n_3 < n_1$$

Where:
n_1 = Number of type 1 links
n_2 = Number of type 2 links
n_3 = Number of type 3 links

$$N_c = 2n_1 + n_2$$

$$L = n_1 + n_2 + n_3$$

So:

$$n_2 \cong 2L - 2n_3 - N_c$$

$$n_1 \cong N_c - L + n_3$$

So:

$$N_A = \frac{N_c}{(N_c - L)}$$

But:

$$L \cong -\frac{N_F}{2} \ln \frac{N_F - N_c}{N_F}$$

So:

$$N_A \cong \frac{1}{1 + \frac{N_F}{2N_c} \ln \left\{ \frac{N_F - N_c}{N_F} \right\}}$$

For: $N_F > N_c$

This equation is plotted on Figure 43 for $N_F = 100$. Agglomeration becomes appreciable when $\frac{N_c}{N_F} > .6$ (i.e. 60% of the file is joined at least once).

Using the data of Figure 36 to relate N_c to D and the above equation to relate N_A to N_c results in Figure 44, on which is superimposed the data of Figure 16. The above equation fits the data very well up to $N_c/N_F \approx .75$. Above that level, the number of n_3 type joins that do not unite clusters becomes appreciable, and a more exact treatment is required (based on resolving the two possible kinds of type 3 joins). The simple equation, however, is sufficiently accurate to serve as a guide to system design.

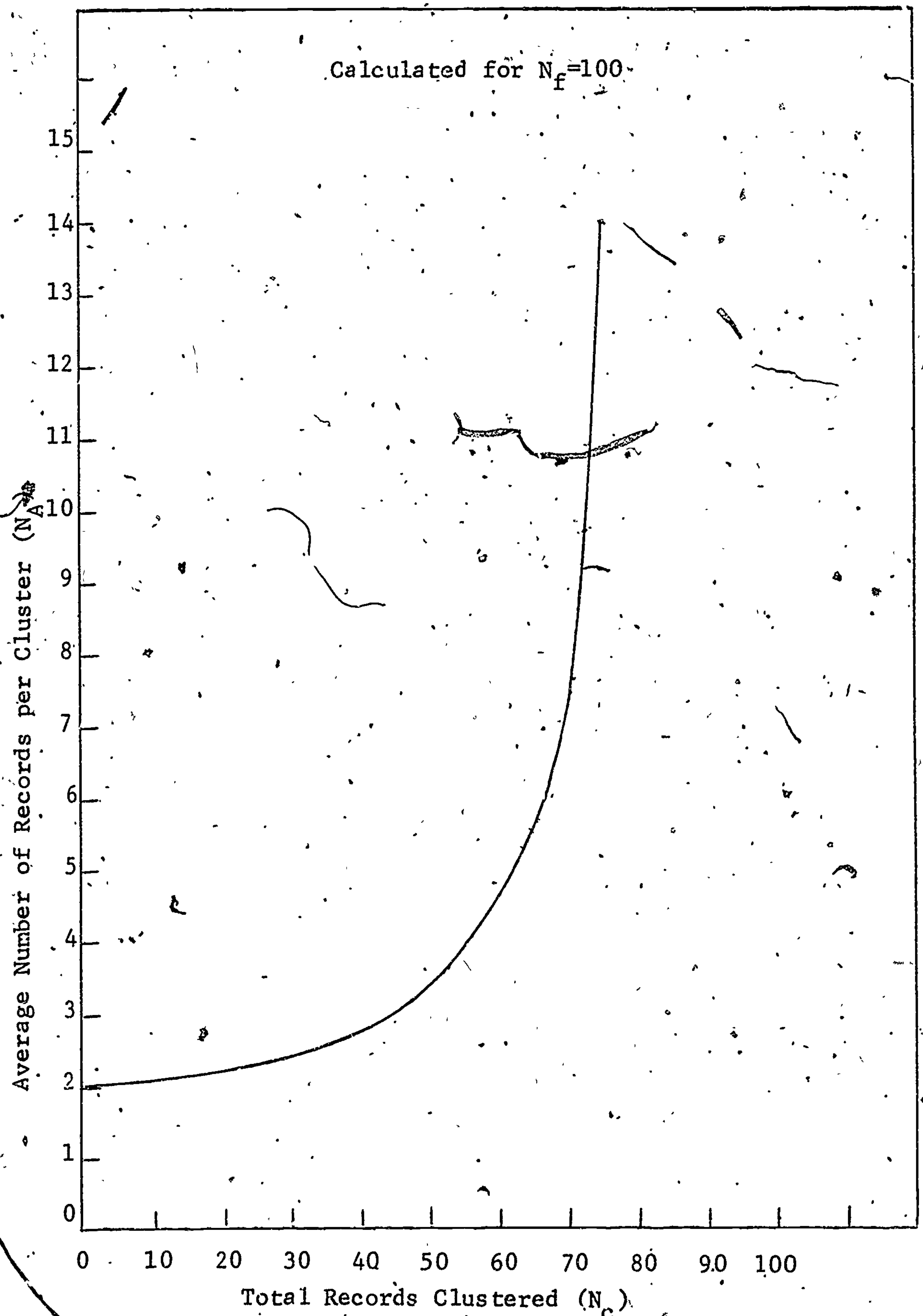


Figure 43. Agglomeration vs Number of Records Clustered

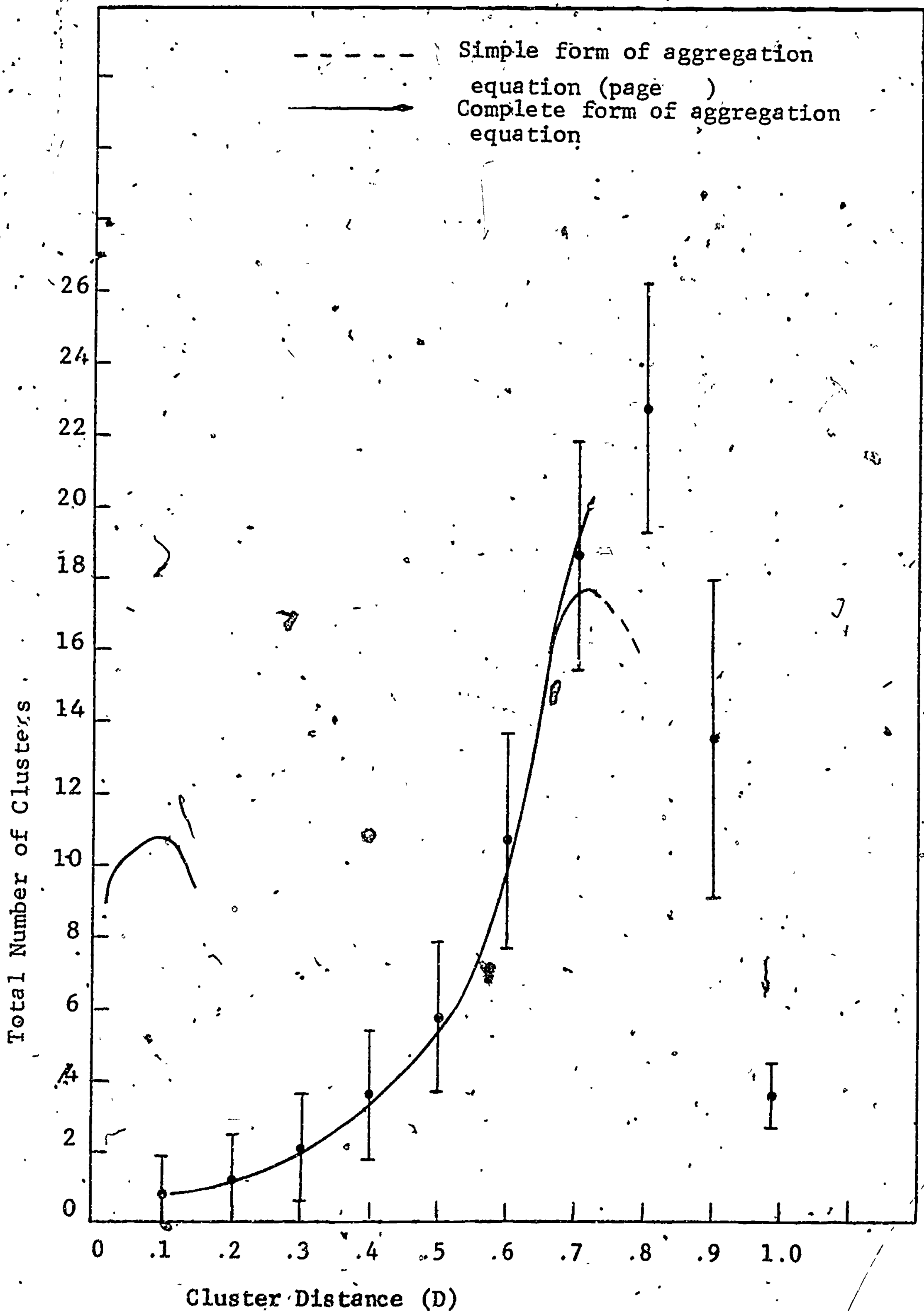


Figure 44. Number of Record Clusters vs Cluster Distance

STATISTICAL MODEL OF THE ACCURACY OF CLUSTERING RECORD ASSIGNMENT

The procedure used in evaluating a cluster for experiments 2 and 3, wherein each record belongs to one of two classes (relevant vs. non-relevant or CACon Section X vs. CACon Section Y) is to total the number of records of each type within a cluster, and assign the cluster to whichever class has a majority. For instance, if a cluster contained 10 records, of which 7 were relevant and 3 were non-relevant, the cluster would be designated relevant, 7 assignments would be counted as correct, and 3 would be counted as errors. However, it is not correct to deduce from this data that the accuracy of clustering record assignment is 70%. Rather, the assignment performance must be compared with the frequency with which correct assignments would be made by chance alone. For the case of a 10-record cluster, no more than 5 incorrect assignments can be made. In other words, even if records were assigned to clusters on the basis of chance, because clusters are labeled as being type A or type B based on their majority constituents, no more than 5 incorrect assignments could be made to a 10-record cluster. A more detailed examination of the statistics shows that the average chance level is somewhat greater than the minimum. Recall that for the experiments designed, there were always equal numbers of the two kinds of records in the set to be clustered, so that the probability that a given record is either one type or another is .5.

For a 2-record cluster, there are 4 possible combinations of records:

Combinations	Score
++	2
+-	1
-+	1
--	2
4 - Total Combinations	6 = Total Score

Since each combination is equiprobable (approximately), the average score attained by chance for a two-record cluster is 1.5 (i.e. 6÷4). Similarly, for a 3-record cluster, there are 8 combinations:

Combinations	Score
+++	3
++-	2
+-+	2
-++	2
--+	2
-+-	2
+--	2
---	3
8 Total Combinations	18 = Total Score

For this case, the average score attained by chance alone is $\frac{18}{8} = 2.25$.

Calculating the chance levels for progressively larger clusters leads to the curve shown in Figure 45. As is shown on that figure, the relationship between the average score attained by chance and the cluster size is approximately linear, and may be estimated reasonably well by the equation:

$$N_R = .625N_c + .30 \text{ for } N_c \geq 2$$

This equation may be used to calculate the extent to which a given set of clusters exceed the chance level in the accuracy of their record assignments.

The score attained by a cluster run is calculated as the fraction of total assignments that are correct, above the chance level (S). At any given cluster distance, the number of records clustered (N_c) and the number of clusters (N) are tabulated, so that the average number of records per cluster (N_A) is:

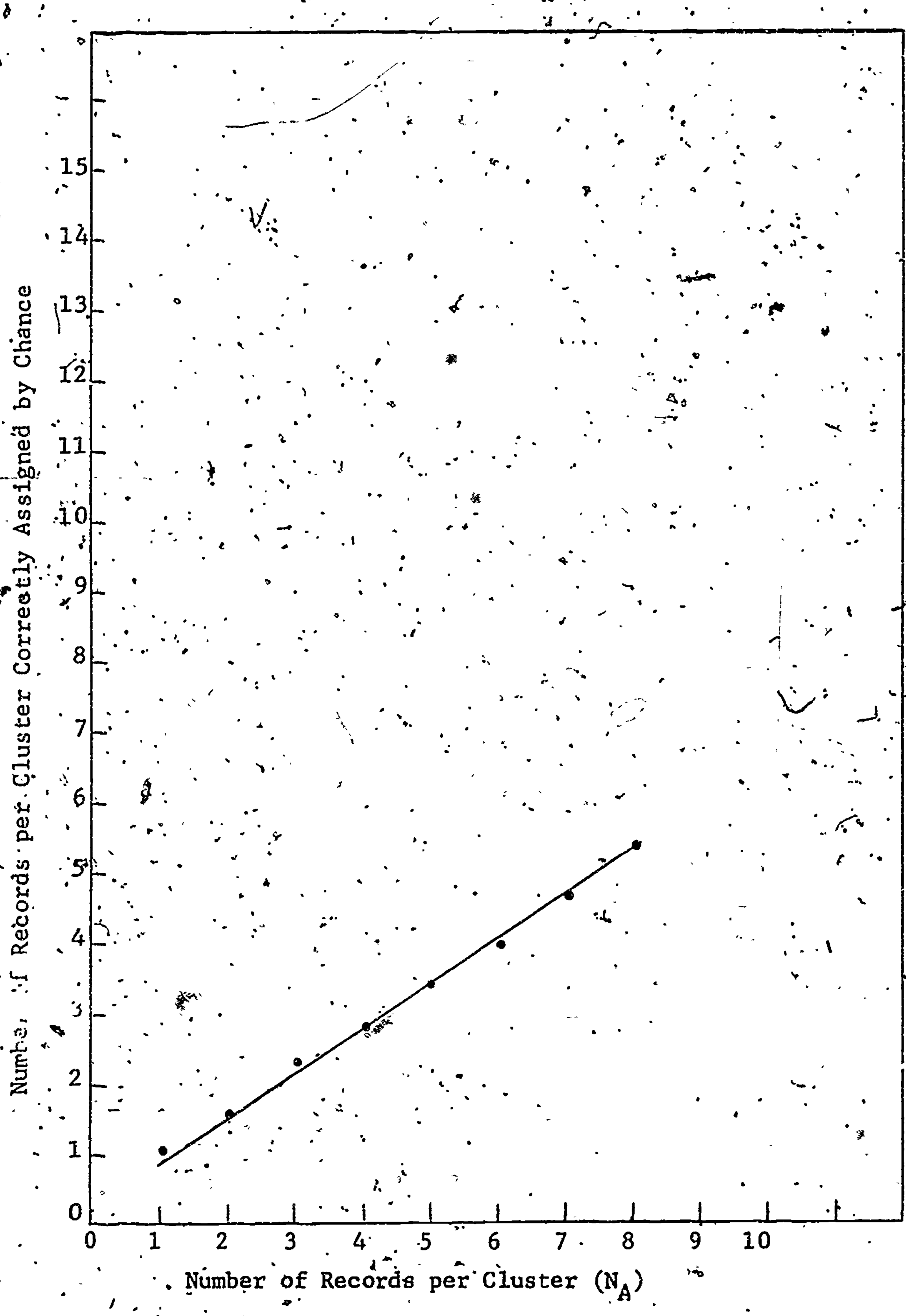


Figure 45. Correct Cluster Assignments by Chance vs Agglomeration

$$N_A' = \frac{\overline{N}_c}{N}$$

The chance level of correct assignments for a cluster of size N_A is given by the N_R equation evaluated at $\frac{\overline{N}_c}{N}$

$$N_R = .625 \frac{\overline{N}_c}{N} + .30$$

So, the total number of correct assignments, by chance alone (N_{RC}) is the number of correct assignments per cluster times the number of clusters

$$N_{RC} = N_R N = .625\overline{N}_c + .30 \cdot N$$

So for N_R total correct cluster assignments, the score is given by

$$S = \frac{N_R - N_{RC}}{N_A - N_{RC}}$$

S has the properties that S=0 if the assignments are correct only at the chance level

S=1 if the assignments are all correct

$1>S>0$ if $N_A>N_R>N_{RC}$ and S is linear with N_R.

Applying this formula to the data on Figure 13 leads to Figure 46. It is clear from this figure that the accuracy with which simple clustering makes record assignments to clusters is very substantial (aboue 80%) for cluster distances less than .5, but that at larger distances it rapidly falls off to unacceptably low values. This is not surprising. If two records have 50% or more of their terms in common, it is not surprising that they should be grouped together. Also, if two records have only about 20% of their terms in common, it is not surprising that grouping is little better than chance.

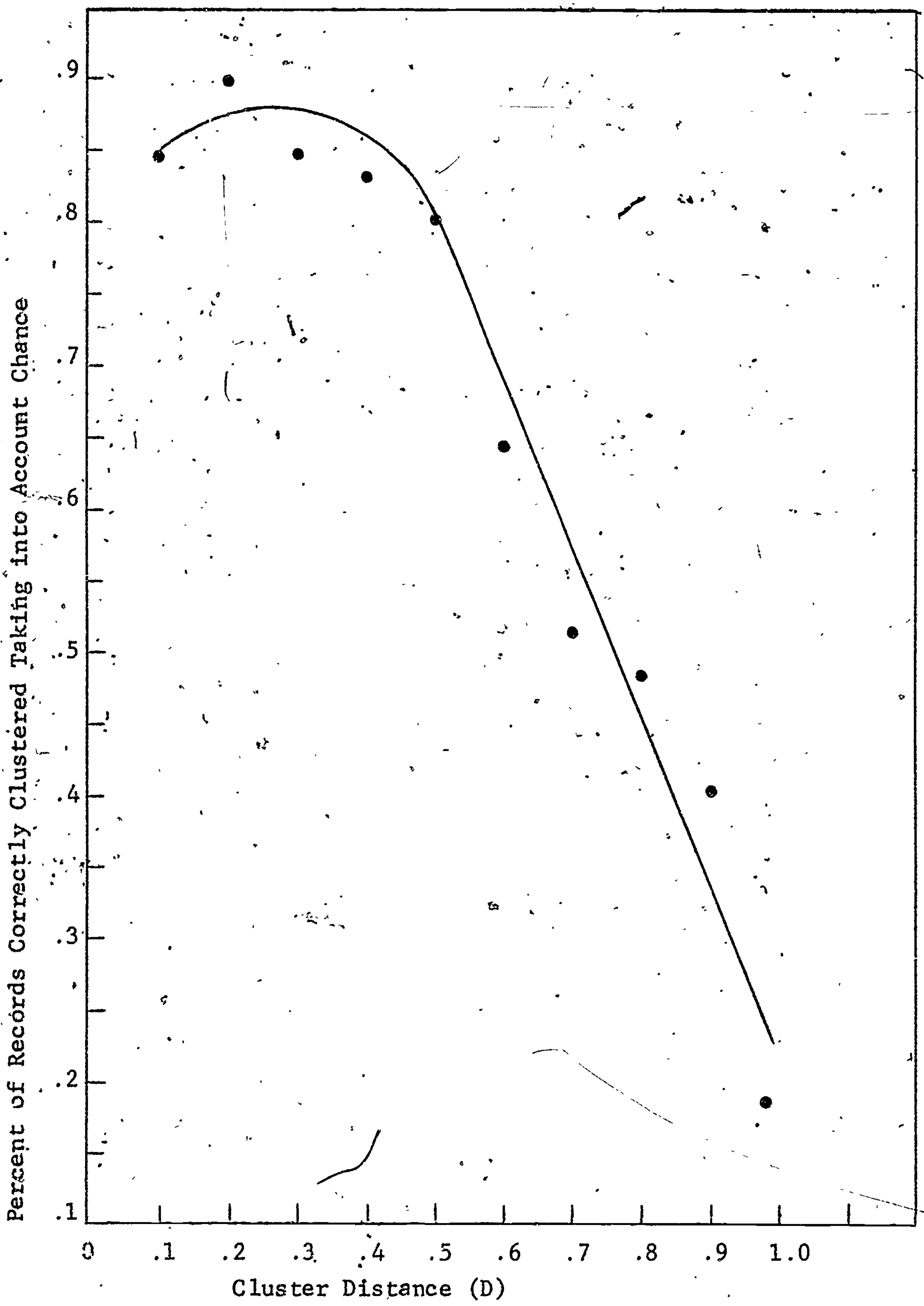


Figure 46. Correct Cluster Assignments (Allowing for Chance) vs Cluster Distance

At a distance of .5, only about 13% of records are clustered and the average cluster size is only about 2.7 records per cluster, so that the number of user decisions have only been reduced from 100 to about 94 (i.e.

$$N_F - (N_A - 1) \cdot \frac{N_c}{N_A} = \text{the number of user decisions required}).$$

This performance is not significantly beneficial to the user. This analysis reemphasizes the need to incorporate semantic information into the system in order to increase S at larger values of distance, where the reduction in the number of required user decisions is more significant.

Examination of individual runs shows that the primary reason for incorrect groupings is the failure of semantically related but non-identical strings, such as greyhound and dog, to match. This is a problem that cannot be solved by a change in the choice of clustering distance measure, because changing the measure cannot recapture the semantically buried information. Rather, a means is needed to record the conceptual relatedness of terms. ATC is an approach to this end using statistically constructed intellectual term classes. Because the term classes always map terms into groups with larger values of N_j, the mapping is subject to the criticism that it sacrifices percision for recall. That is, Salton has conjectured that it is the intermediate frequency terms that are the most important for information retrieval[30]. The very low frequency terms, it is argued, cannot be very important because they cannot participate in many matches. Also, the very high frequency terms cannot be very significant because they lack specificity, i.e. they match so often that the information value of a match is small. Accordingly, he recommended that very low frequency terms be grouped into intermediate frequency classes, and very high frequency terms be divided up into intermediate frequency term phrases. These suggestions seem unassailable in the context on one-step searching. Yet, in the context of multi-step searching, it seems preferable to use the structured vocabulary methods described in Experiment 4. Representing terms within such an hierarchy allows for the matching to be performed within the limitation of a given range of concepts (the idea of SBC), and to match strings that are not identical with a match value less than unity, and to perform the matches at selected levels of generality. The process of adding to a term the names of the categories in which it is found is to carry with the term the context of its use. Williams found this kind of information useful in directing a user query to an appropriate data base[31]. It is just this kind of information that is used implicitly in dialog to break the ambiguity of term definition. Thus whereas "absorption" has two distinct definitions (at least) they may be disambiguated by noting that one is in the spectral sense and one is in the physical sense. The use of a formalism in which the specific term mappings are associated with a term occurrence suggests a natural interface with artificial intelligence processing tasks. Using AI techniques, perhaps terms can be disambiguated by consideration of the contexts in which they occur. Similarly, the occurrences of the labeled term suggests that the identification of the contexts would be made easier as well, perhaps through local consensus.

The effects of vocabulary mapping can be evaluated in terms of the statist~~ical clust~~ering model. Every word in the language is a precise instrument, and any time virtually any term is replaced with another, meaning is changed. Any time that meaning is degraded, the <u>accuracy</u> with which records can be grouped is depressed. Of course, if terms are replaced by more general terms, ΣN_j^2 is increased and the probability of match is increased, so that coverage and agglomeration are increased. The experiments performed suggest that for accuracy to be sufficient, coverage must approach 100% at a distance of less than about .5. Convenience would suggest that average cluster size should be about $\frac{N_F}{4}$ at that distance as well. The statistical model predicts that these conditions would require $\sum_j N_j^2 \sim 13{,}500$ for a file of 100 records. The actual value of $\sum_j N_j^2$ in the experiments is about 5100.

Rough calculations show that ATC can achieve the factor of about 3 that is required to raise $\sum N_j^2$ to the projected feasible range. By increasing the number of links between records, ATC can be projected to achieve resolution of relevant and non-relevant records to a degree that is useful to a user. However, this projection should be regarded only as a motivation for further work, and not as a guarantee of success.

PROCESSING COST

The costs involved in applying simple clustering to about 100 bibliographic records from either CACon or Ei COMPENDEX include identifying the terms, applying a stop list, utilizing controls and, finally, clustering. In experimental runs, on an IBM 370/158, these steps consume about 20 cpu seconds for the term preparation and 20 cpu for the clustering. In production runs, the computation time would be considerably less. Much of the term identification process could be saved by preprocessing the records (i.e. storing stems and stop-listed terms, perhaps in a canonical form). The clustering time could also be greatly reduced. The experimental runs gave much more detail than would be required by a user. Perhaps 15 cpu seconds would be a reasonable estimate for 100 records and about 60 cpu for 1,000 records.

The ATC term mapping requires about 300 cpu seconds for two issues of CACon. This is the cost for associating a term with a subsection, section and supersection. Several hundred more seconds are required to restructure the data base to put it into a form to take advantage of mapping.

The SBC clustering should cost less than the simple clustering because fewer terms per record are accepted by the content focusing mechanism. However, firm cost estimates are not available yet for SBC.

The ATC term mapping, structuring and labeling operations are done only once on a data base and are then available for all searches. In essence, global information is processed once, saving each separate user from repeating the same intellectual operations.

5. DISCUSSION

How is an IR system to be made efficient? For string processing programs, the historical first step was to save on the number of string compares required during single retrieval. Inverted files handle that phase very well by sorting the file into a structure such that the anticipated question, "Where does string 'xxxx' occur?" is answered for all strings before any searches are done. This saves each user the cost of doing that sort separately. On a somewhat more sophisticated level, ATC similarly saves each user from analyzing the context of each term by using global statistical information to relate all the implicit context definitions before any searches are done. That is, just as string processing programs save on comparisons by comparing only those strings for which a match is possible (based on a crude first approximation such as LCB[32]) semantic processing programs should save on compares by using a crude first approximation to meaning (such as ATC).

When one projects the structure and capabilities of the IR systems of the future, one is inevitably drawn to consider the automation of semantic and cognitive processes. (i.e. the functions performed by an ideal librarian.) In this regard, one is led to ask, "What is the future role of current statistical string processing procedures in future systems that will be doing semantic processing?" It is tempting to think that the future IR system would be a "world brain" in which statistical processes had no place, i.e. where new information was folded into an existing knowledge bank by a process analagous to "understanding". In such a circumstance, one might assume that retrieval would be very fast, analagous to the human power of abstraction of concepts. However, there are two problems with this point of view. First, even for humans, recall is statistically based. Frequently used information is easily retrieved in the human mind while infrequently used information is often remembered only with great difficulty. Moreover, such performance is reasonable. First, when memory space is finite, and response time is important, it makes sense to put the highest priority records in the most accessible places. The second problem is that the process of understanding generally means developing the capacity to answer a given class of problems by preprocessing the data. For example, if I'm told that "John is in Texas". I can easily answer the question "Where is John?" However, there are many other questions such as "Why is John in Texas?", that are not easily anticipated nor are they easily handled by standard (canonical) forms. Such questions may require inference and the use of implicit information. The point is that the large number of questions that may be asked about text is large, perhaps infinite, and no system can be expected to have answered all of them on a preprocessor basis. Some large classes of questions may be answerable on a preprocessor basis (like "Where is John?"), but many of the unanticipatible questions will require run time analysis of records. To be efficient, it seems that the two kinds of questions (high frequency anticipatible or low frequency unanticipatible) should use memory in different ways. The ATC and sequential search formalism has an obvious extension that seems to accomodate these two needs. It consists of the representation of each term by an n-tuple in which each field corresponds to an attribute and each entry corresponds to a value. For a multi-step system based on such a representation, the Boolean search component would access a limited range of fields, intermediate processing would have access to more fields, and semantic processing would have access to all fields. Such representation has not been the focus of recent AI research activity because of the apparent storage economies and the other successes achieved by semantic nets and linked lists. In the use of these methods, every attribute is a node. What is suggested here is that the nodes in a semantic net need not be bare character strings. Rather, they may be n-tuples. Then the semantic net becomes a network between n-tuples. That is, one of the most serious problems in latural language AI is the prioritization of computer processing tasks. Processing demons are one attempt[12]. Perhaps the high frequency memory access needs would be best met by explicit n-tuple representation while the low frequency needs would be met by pointers and semantic net relations.

Reaction time experiments[33] suggest that human memory works on a bucket principle such that weak relations identify the bucket in which the words that are candidates for a given usage are stored. Intellect is then required to examine the contents of a given bucket and to select the appropriate word. It is interesting to note that if the n-tuple representation of terms were used as the bucket forming mechanism, and if each entry in the n-tuple were binary, for n=20, there would be enough possibilities to disambiguate 10^6 words. The 20 bit strings would allow classes of words with similar meanings to be retrieved directly through their similar bit strings. That is, the 20 bit strings could provide a fast bucket retrieval mechanism for the content addressibility of terms. It may be co-incidental, but in the game of 20 questions, 20 binary responses to a more-or-less standard collection of questions is sufficient to disambiguate (guess) the selected thing (word) from a collection of possibilities of the order of 10^6.

The overall point is that multi-step processing of records consisting of terms, each of which is represented by augmented fields (n-tuples), some of which are statistical in origin and some of which are semantic, seems to suggest a system design that can accomodate levels of processing from simple record retrieval to detailed AI. This work has demonstrated the value of multi-step processing at the statistical end of the spectrum wherein practical application to traditional IR problems may be imminent involving user of the ATC or related methods. Moreover, it is suggested that application and interfacing of these methods with those in the realm of semantic information processing seems warranted, to tackle the IR problems of the future.

APPENDIX A

LIST OF ALL SYMBOLS USED

$D(a,b)$	=	The distance between a and b as specified by a given measure. The distance may also be called "D" when a and b (or their equivalents) are specified in another manner.
$f(L,N_F)$	=	The link redundance factor = the number of records clustered by L links for a file of N_F records.
J	=	The number of unique terms in a file.
L	=	The number of links.
N	=	The number of clusters in a file.
$N(R_i,R_j)$	=	The number of pairs of records in a file.
$N_A(x)$	=	The number of records in a cluster.
N_c	=	The number of records clustered.
N_F	=	The number of records in a file.
N_j	=	The number of records in a file in which the jth term ($1 \leq j \leq J$) occurs.
N_R	=	The number of records clustered correctly.
N_T	=	The number of terms in a record.
$P(k)$	=	The probability that two records have at least k terms in common (i.e. k term matches).
R_i	=	the i'th record in a file ($1 \leq i \leq N_F$).
S	=	Accuracy of clustering record assignments, allowing for statistics of chance.
T_j	=	the jth term in a file ($1 \leq j \leq J$).
accuracy	=	the fraction of clustered records that are assigned to clusters correctly.
agglomeration	=	the average number of records per cluster at a given distance.
ATC	=	Automatic Term Classification

coverage	=	the number of records in a file that are clustered at a given cluster distance.
document	=	a publication or piece of one.
field	=	a subdivision of a record i.e. author field, title field, etc.
precision	=	the fraction of records retrieved that are relevant.
recall	=	the fraction of relevant records in a data base that are retrieved.
record	=	the representation of a document in a data base, usually consisting of author, title, CODEN, and source fields.
n_1	=	the number of type 1 record links (i.e. the number of new links between previous unlinked records.
n_2	=	the number of type 2 record links (i.e. the number of new links between previously unlinked records and linked records.
n_3	=	the number of type 3 record links (i.e. the number of new links between previous linked records.
f_1	=	the largest fraction of normalized term occurrences, for a single term, in any CACon division (subsection, section or supersection).
f_2	=	the second largest fraction of normalized term occurrences, for a single term, in any CACon division (subsection, section or supersection).

Simple Clustering	=	clustering of records without any preprocessing of the terms that they contain.
SBC	=	Subset Based Clustering. A clustering technique using term classes derived from statistical preprocessing to accomplish three functions: degrees of term matches, term disambiguation, and restriction of scope of attention.
$\overline{L}$	=	the average number of links per record pair.
$\underline{P}(k)$	=	probability of at least k term matches between two given records.
p(j)	=	probability of a match on the jth term for two given records.
$\underline{P}$(ex k)	=	probability of exactly k term matches between two given records.

APPENDIX B

TERM MATCHING EQUATIONS

N_F = The number of records in the file.

N_j = The number of records with term j, $N_j \leq N_F$

$\underline{P}(k)$ = Probability of at least k term matches between two records.

$\underline{P}(0)$ = Probability of no term matches between two records.

$p(j)$ = Probability of a match on the j'th term.

$\underline{P}(ex\ k)$ = Probability of exactly k term matches between two records.

$p(not\ j)$ = Probability of no term match on the j'th term.

L = Total number of links in the file =

$$\sum_j \frac{N_j(N_j-1)}{2} = \sum_j \binom{N_j}{2}$$ = the number of pairs of identical records.

$\overline{L}$ = The average number of links per record pair =

$$\frac{L}{\text{total number of record pairs}} = \frac{\sum N_j(N_j-1)}{N_F(N_F-1)}$$

$p(j)$ = (probability jth term is in R_1) $\cdot$ (probability that jth term is in R_2 given that it is in R_1)

$$p(j) = \frac{N_j}{N_F} \cdot \frac{N_j-1}{N_F-1}$$

$$\underline{P}(ex\ 0) = \prod_{j=1}^{J} p(not\ j) = \prod_{j=1}^{J} \left(1 - \frac{N_j}{N_F} \cdot \frac{N_j-1}{N_F-1}\right)$$

$$\ln \underline{P}(ex\ 0) = \sum_{j=1}^{J} \ln p(not\ j) = \sum_{j=1}^{J} \ln\left(1 - \frac{N_j}{N_F} \cdot \frac{N_j-1}{N_F-1}\right)$$

for $N_j << N_F$.

$$\ln \underline{P}(\text{ex } 0) = -\sum_{j=1}^{J} \frac{N_j(N_j-1)}{N_F(N_F-1)} - \sum_{j=1}^{J} \left\{\frac{N_j(N_j-1)}{N_F(N_F-1)}\right\}^2$$

$$\ln \underline{P}(\text{ex } 0) = -\bar{L} - \sum_{j=1}^{J} \left\{\frac{N_j(N_j-1)}{N_F(N_F-1)}\right\}^2$$

$$\underline{P}(\text{ex } 0) = \exp\left\{-\bar{L} - \sum_{j=1}^{J} \left\{\frac{N_j(N_j-1)}{N_F(N_F-1)}\right\}^2\right\}$$

Usually $\frac{N_j}{N_F}$ is so small that it is a good approximation to take

$$\underline{P}(\text{ex } 0) = \exp -\bar{L}$$

If $N_j \simeq N_F$ for only one j, (denoted "0"), then

$$\underline{P}(\text{ex } 0) = \exp -(\bar{L} + f_0^2) \text{ for } f_0 \equiv \frac{N_0(N_0-1)}{N_F(N_F-1)}$$

When all $\frac{N_j}{N_F} << 1$

$$\underline{P}(1) = 1 - P(\text{ex } 0) = 1 - \exp -\bar{L}$$

$$\underline{P}(2) = \underline{P}(1) - \underline{P}(\text{ex } 1)$$

$$\begin{aligned}\underline{P}(\text{ex}1) = \; & p(1) \cdot p(\text{not } 2) \cdot p(\text{not } 3) \cdots p(\text{not } J-1) \cdot p(\text{not } J) \\ + \; & p(\text{not } 1) \cdot p(2) \cdot p(\text{not } 3) \cdots p(\text{not } J-1) \cdot p(\text{not } J) \\ & \vdots \\ + \; & p(\text{not } 1) \cdot p(\text{not } 2) \cdot p(\text{not} 3) \cdots p(J-1) \cdot p(\text{not } J) \\ + \; & p(\text{not } 1) \cdot p(\text{not } 2) \cdots p(\text{not } J-1) \cdot p(J)\end{aligned}$$

$$P(\text{ex } 1) = \sum_{j=1}^{J} \frac{p(j)}{p(\text{not } j)} \cdot p(\text{not } 1) \cdot p(\text{not } 2) \cdots p(\text{not } J)$$

$$= \underline{P}(0) \sum_{j=1}^{J} \frac{p(j)}{p(\text{not } j)}$$

$$= \underline{P}(0) \cdot \sum_{j=1}^{J} \frac{N_j(N_j-1)}{N_F(N_F-1)} \Bigg/ \left\{1 - \frac{N_j(N_j-1)}{N_F(N_F-1)}\right\}$$

$$\simeq \underline{P}(0) \sum_{j=1}^{J} \frac{N_j(N_j-1)}{N_F(N_F-1)}$$

$$\simeq \underline{P}(0) \cdot \overline{L}$$

$$\simeq \overline{L} \cdot \underline{P}(0)$$

So $\underline{P}(2) = 1-e^{-\overline{L}} - e^{-\overline{L}} \cdot \overline{L}$

and in general, for all $N_j << N_f$

$$\underline{P}(k) = 1 - \sum_{k=0}^{k-1} \frac{\overline{L}^{-k} e^{-\overline{L}}}{k!} \qquad k \geq 1$$

APPENDIX C

The software developed for this project is based largely on programs previously developed at IITRI, including the file inversion software and several clustering programs. In order to conduct the experiments, software modifications were made and a few special purpose programs were written. These programs are briefly described below.

Standard Computer Search Center (CSC) file inversion software extracts terms from specified fields of each record (usually title and keyword fields) in a file and associates with each one the number of the record (posting) in which it occurs. Small modifications of this procedure allowed different identification to be associated with each term occurrence. The most useful choice was the CACon Section-Subsection Number. The result of the INVERT program is a file where each record consists of a term of up to 20 characters followed by a 6-character CACon Section and Subsection number.

This file is sorted on the term string within each block of entries for a single term. The entries are sorted on the Section/Subsection number field. This procedure places all occurrences of a given term together, and orders the occurrences according to Section/Subsection numbers.

After the sort is completed, multiple occurrences of any term in any Subsection will be stored consecutively. Next, the multiple record occurrences of each term are counted and deleted. Then a new record is created which consists of the term string followed by a list of all of the postings for that term. The CSC SQUEEZ program was modified to accomplish these ends. SQUEEZ creates, for each term, one or more varying length blocks each containing up to 100 separate posting locations. Each of these posting locations can accomodate all of the term postings within a given category (CACon divisions). That is, if a term occurred in up to 100 separate Subsections, then only one record would be needed; if between 101 and 200, then 2 records, etc. Each block of up to 100 separate posting locations contains the number of postings in that block, the first 20 characters of the term, the number of blocks created for the term so far, and the string of pairs consisting of the Section and Subsection numbers and the frequency of occurrence within that section. This file format was chosen to facilitate statistical calculations of term correlations with CACon divisions (Supersections, Sections and Subsections).

The first step in analyzing the inverted file is to normalize the term frequencies for each section, to allow for the variance in the size of the sections. This normalization was based on the number of terms occurring in each section. Inputs to NORM (the program that performs the normalization) total term frequencies per section and the file created as a result of the SQUEEZ program. A normalizing factor is calculated by dividing the section with the most terms by the number of terms in each section. A table of these normalizing factors is created. The file is read through term by term multiplying the frequency of the Section by the appropriate normalizing factor in the table. The results are written into a new file using the same structure they were read from.

The file created by SINORM is used in the second step (S2SEC) to find the sections where the first and second peaks exist for a given term. For each term, the string CACon Section number and corresponding normalized frequencies are read (the subsection data are combined into section groups) and the sections with the highest frequencies are identified and printed out. Four 100 position arrays are declared to keep track of where these peaks have occurred. Each term increments a position in one array for the first peak and another array for the second peak. There are 2 separate arrays declared for high frequency (more than 25 normalized occurrences) and low frequency (less than 25 normalized occurrences) terms. These arrays are printed out at the end of the run.

Slight modifications were made to the second step in order to obtain information on the first and second peaks within CACon Supersections. In S2SUPER, the normalized section and subsections are combined into supersection groupings and the peaks are printed out as before. The four arrays are similarly incremented and printed out.

In order to examine these peaks for subsections, the file must be re-normalized based on term frequencies of the subsections. The file created by SQUEEZ is used as input to SINORM2 which, along with S2SUB, produces output similar to the other versions of steps 1 and 2. Figure C-1 summarized this entire procedure.

The user relevance experiments and the programs used for it, required the setting up of files with certain types of evaluated records. The record numbers of the citation satisfying a users profile as well/or whether it was denoted as relevant or nonrelevant by the user was keypunched. With this information the standard utility program, SELECT, could retrieve these records from tapes maintained at IITRI containing the entire citations. These tapes of citations are organized by volume and record number and contain records in a standard internal format. The citation numbers of interest and the file of complete records for a given volume, serve as input to SELECT. These two inputs are sorted into record number order so that these files may readily be compared for matches. When two record number match, the corresponding citation is written out to a new file. Appropriate selection of the input citation numbers results in the creation of a file of 50 relevant and 50 non-relevant complete citations for a given user. The file created by SELECT is next processed by EXTRACT. EXTRACT organizes the term lists into a form convenient for clustering. Next, the subroutine TERMER is called. TERMER has 3 relevant parameters: a pointer to the citation, the fields to be analyzed (CODEN, title, etc.), and a character string of run time parameters that specify options such as inclusion of single occurrence terms in the distance measure and output format.

Terms are extracted by the subroutine TERMER. Under the direction of EXTRACT, TERMER creates a set of lists of all terms found, their numbers of occurrences and the locations of those occurrences.

The file created by EXTRACT is used by the program CLUSTER. First the Document-Term array is read and stored in a reduced form. Calculations are performed for term distances. The resulting cluster analysis is printed out as a function of the distance value in dendograms and other data summary formats. Flow charts for the interaction of these procedures are shown on the following pages. Computer listings for the major procedures follow subsequently.

TERM MAPPING

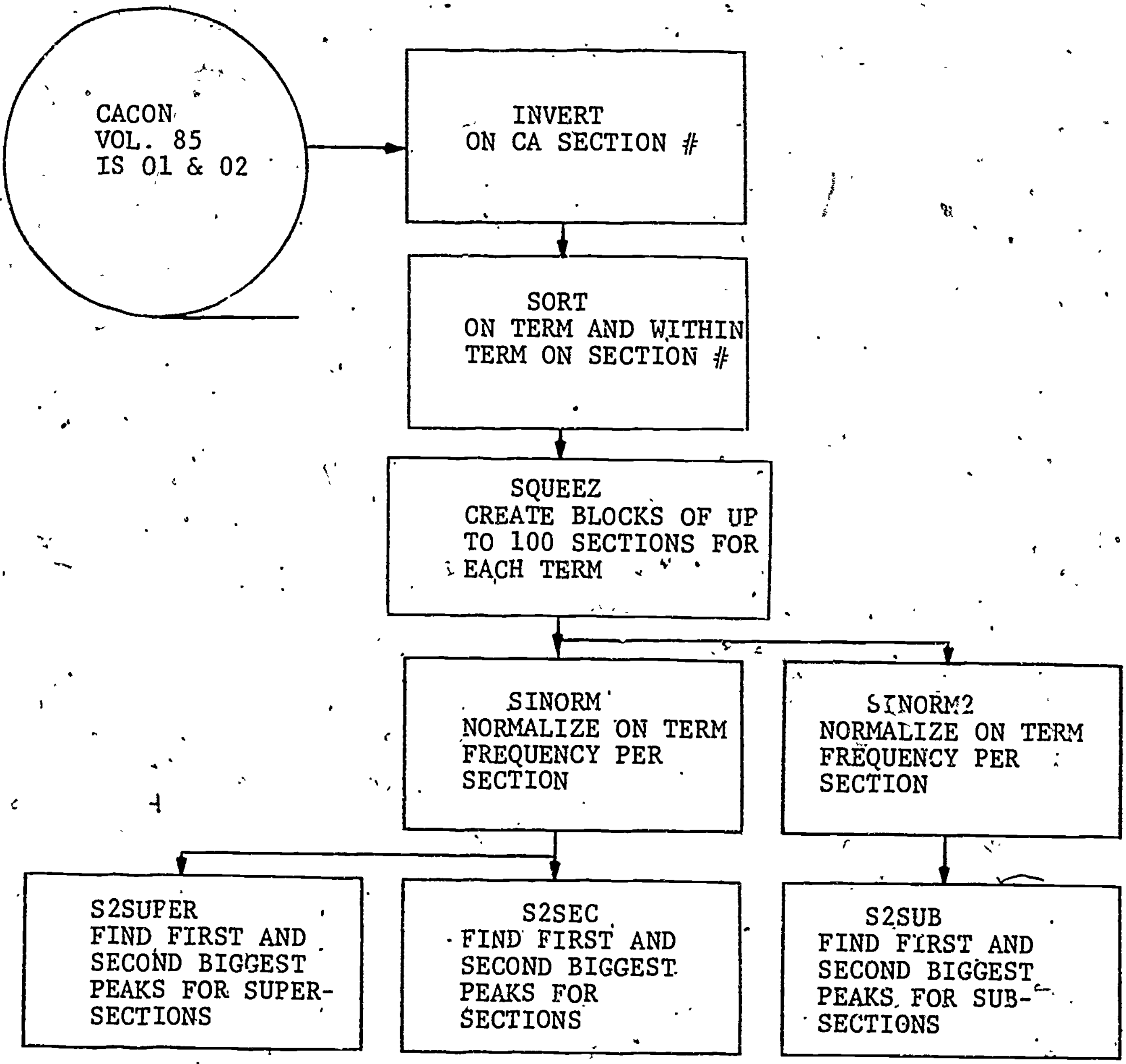


Figure C1. Processing Flow for Experiment 4

USER RELEVANCE

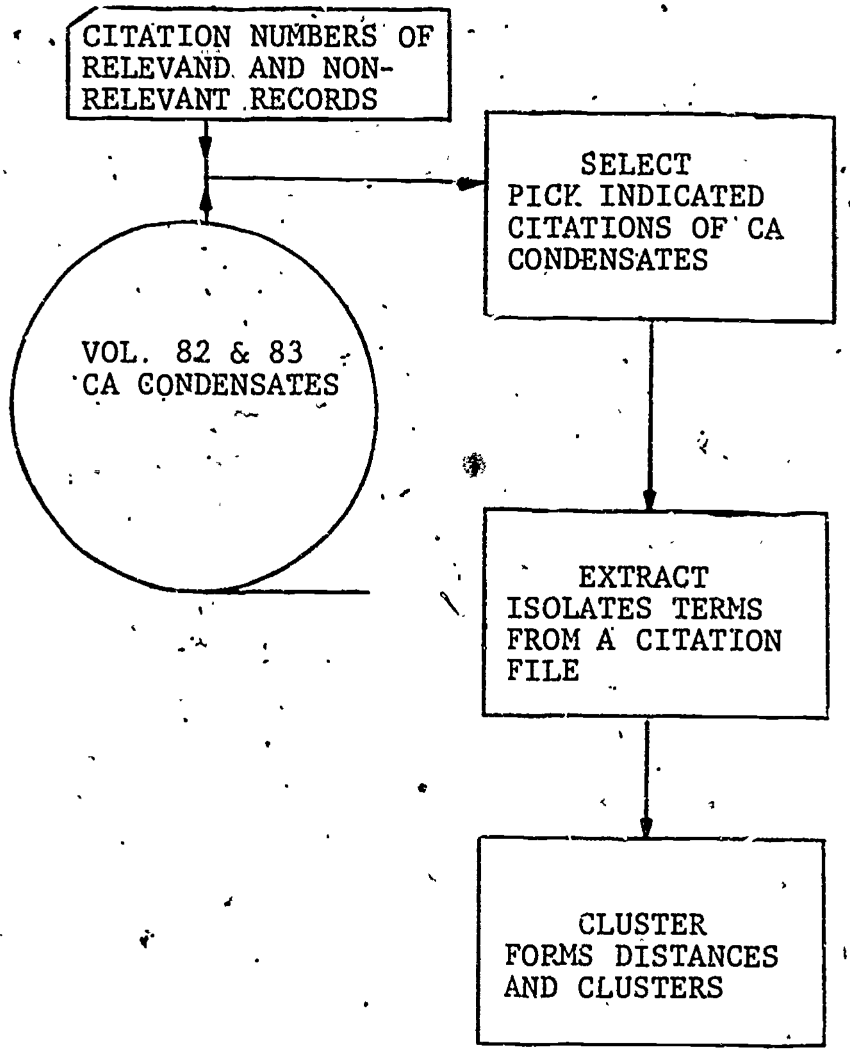


Figure C2. Processing Flow for Experiment 3

```
                                        /* LAST UPDATE: 750103 */
    EXTRACT : PROC (RTP) OPTIONS(MAIN);
/* THIS PL/1 PROGRAM EXTRACTS TERMS FROM CITATIONS AND DROPS
   SINGULAT TERMS.  INPUT IS P.L.S. FO MAT RECORDS OR OPTIONALLY A
   HIT-FILE.  OUTPUT IS TERM LISTS FOR DOCUMENTS & A DICITIONARY.
   THIS EXTRACT DROPS ALL SINGULAR TER'S & ASSIGNS A ZERO IN THE TERM
   LIST                                            */
DECLARE
       NMAX FIXED BIN STATIC, /* NUMBER OF RECORDS TO BE READ */
            ALL BIT(1) ALIGNED STATIC,
        1 HIT_REC,
          2 PROFNUM CHAR (10),
          2 HIT_WT FIXED DEC (5),
          2 ABSNO CHAR (11),
          2 SORT_FLD CHAR (45),
          2 HIT_LIST CHAR (79),
        1 CIT_REC BASED (P),
          2 UNO CHAR (11),
          2 REFNO CHAR (11),
          2 PAD CHAR (1),
          2 DAT FIXED BIN,
          2 LOD FIXED BIN,
          2 LOP FIXED BIN,
          2 DIR (1),
            3 TYPE CHAR (4),
            3 STRT FIXED BIN,
            3 LEN FIXED BIN,
        PROF CHAR (10),
        (PA,PB) POINTER,
        FIELDS (4) CHAR (4) INIT ((4)(4)' '),
        NUM FIXED BIN,
         RTP CHAR (100) VARYING,
       RDP CHAR (100) VARYING,
        HITFILE  FILE RECORD SEQUENTI L INPUT,
        CITFILE  FILE RECORD SEQUENTI L INPUT,
        1 ABSNO1 DEF  ABSNO,
          2 AB1 CHAR (4),
          2 PAD CHAR (1),
          2 AB2 CHAR (6),
        1 REFNO1 BASED (P),
          2 GARB CHAR (10),
```

```
          2 REF1 CHAR (4),
          2 PAD CHAR (1),
          2 REF2 CHAR (6);
        IPH=1;
       RDP=RTP;
       N=0;
        ON ENDFILE(CITFILE) GOTO DONE:
       ON ENDFILE (HITFILE ) GO TO DONE;
       GET LIST (NMAX); /* CUTOFF*/
       GET LIST(PROF,NUM); /* PROFILE NUMBER, NUMBER OF FIELDS
                              EXTRACTED. IF HITFILE IS NOT USED,
                              THEN PROF WILL EQUAL 'ALL' */
        PUT PAGE EDIT('PROFILE: ',PROF,'CUTOFF: ',NMAX)(SKIP,A,A);
        PUT SKIP EDIT('FIELDS: ')(A);
       GET LIST((FIELDS(I) DO I=1 TO  UM)); /*FIELDS TO BE EXTRACTED*/
        PUT SKIP;
        PUT LIST((FIELDS(I) DO I=1 TO NUM));
            IF PROF='ALL' THEN ALL='1'B;
            ELSE ALL='0'B;
   LOOP1:
        IF ¬ALL THEN DO; /*HITFILE USED. READ UNTIL CORRECT PROFILE
                              FOUND                                  */
             READ FILE(HITFILE) INTO( IT_REC);
       IF PROFNUM¬=PROF THEN GO TO LOOP1;
            END;
       N=N+1;
             IF N>NMAX THEN GOTO DONE:
 LOOP2: READ FILE (CITFILE ) SET (P);
       IF ¬ALL THEN DO; /*READ UNTIL CITATION AND HIT FILES COINCIDE*/
       IF AB1>REF1 THEN GO TO LOOP2;
       IF AB1=REF1 THEN DO;
         IF AB2>REF2 THEN GO TO LOOP2:
         IF AB2<REF2 THEN GO TO LOOP1:
         END;
       IF AB1<REF1 THEN GO TO LOOP1;
       END;
       PA=P; /* POINTER TO CITATION RECORD*/
       PB=ADDR(HIT_LIST); /*POINTER TO POSSIBLY BLANK HIT LIST*/
       CALL TERMER(PA,FIELDS,PB,RDP);
       GO TO LOOP1;
   DONE:
       IF IPH=1 THEN DO; /*BEGIN SECOUND PASS*/
             PUT PAGE;
        IPH=2;
             N=0;
             CLOSE FILE(HITFILE);
             CLOSE FILE(CITFILE);
       PB=NULL; /* SIGNAL END OF FIRST PASS */
       CALL TERMER(PA,FIELDS,PB,RDP);
             GOTO LOOP1;
        END;
       PA=NULL; /* SIGNAL END OF PROCEDURE */
       CALL TERMER(PA,FIELDS,PB,RDP);
       END EXTRACT;
*PROCESS ('ATR,XREF');
       TERMER: PROC (PP,FILDS,PT,RTP):
```

```
      DECLARE
      RTP CHAR (100) VAR,
      NO$SW BIT (1) INIT ('0'B) STATIC,
      NOSSW BIT (1) INIT ('0'B) STATIC,
      SECSW BIT (1) INIT ('0'B) STATIC,
      SECON BIT (1) INIT ('1'B) STATIC,
       (PP,PT) POINTER,
       FILDS (4) CHAR (4),
       1 CIT_REC BASED (PQ),
         2 UNO CHAR (10),
         2 REFNO CHAR (11),
         2 PAD CHAR (1),
         2 DAT FIXED BIN,
         2 LOD FIXED BIN,
         2 LOP FIXED BIN,
         2 DIR (1),
           3 TYPE CHAR (4),
           3 STRT FIXED BIN,
           3 LEN FIXED BIN,
       PH1 BIT(1) ALIGNED STATIC INIT('1'B),
       RECNUM FIXED BIN(31) STATIC INIT(0),
      STRNG CHAR (4000) BASED (PR), /*STRING FOR CITATION RECORDS*/
      STR CHAR (79) BASED (PT), /*STRING FOR HIT RECORDS */
       ALPH CHAR (26) INIT ('ABCDEFGHIJKLMNOPQRSTUVWXYZ'),
       WORD CHAR (20) VARYING,
      WRKSTR CHAR (2000),
      WRKST (1500) CHAR (1) DEF WRKSTR,
       (Q,LAST) POINTER,
       PUNCH FILE STREAM OUTPUT,
      (INDX(2:52) PTR, /*POINTERS TO TERM LISTS*/
        FIRST FIXED BIN INIT (0),
      NW FIXED BIN INIT (0), /*NUMBER OF NON-SINGULAR TERMS*/
      NDX BIN FIXED (15), /*  COUNTER FOR STOPWORD CHECKING          */
      NUMWRDS FIXED BIN INIT (0), /*NUMBER OF UNIQUE TERMS FOUND */
      BADWRD4 CHAR (64) INIT
('WERE WITH REFS MADE THAN THIS THAT SOME SUCH FROM INTO BEEN BOOK'),
       BADWRD5 CHAR (33) INIT ('WHICH STUDY AFTER  THESE  THEIR'),
      BADWRD7 CHAR (16) INIT('PERCEN  BETWEEN'),
      BADWRD9 CHAR (31) INIT ('DISCU SED  DISCUSSES  CONDITION'),
       BADWRD3 CHAR(39) INIT
       ('AND THE FOR HAS ARE WAS NOT ONE USE MAY'))
        STATIC,
       1 REC STATIC,
         2 NUM FIXED BIN INIT (0), / SEQUENTIAL RECORD NUMBER */
         2 ONE FIXED BIN INIT (1), /  ALWAYS SET TO ONE */
         2 KNT FIXED BIN, /*NUMBER OF TERMS IN RECORD */
         2 LIST (100) FIXED BIN,
       1 LTERM BASED (P1), /*STRUCTURE FOR EACH TERM FOUND*/
         2 TERM CHAR (20),
        2 NO FIXED BIN, /*INDICATES IF TERM IS NON-SINGULAR */
         2 CNT FIXED BIN(15),
         2 RECN FIXED BIN(31),
         2 NEXT POINTER;
       RECNUM=RECNUM+1;
       IF PT=NULL THEN DO;
            PH1='0'B;
```

```
        REC.NUM=0;
        RETURN;
    END;
  IF PP=NULL THEN GO TO PRINTR; /*LAST TIME CALLED*/
  PQ=PP; /*POINTER TO CITATION RECORD */
  PR=ADDR(TYPE(LOD+1)); /* POINTER TO CITATION STRING */
  IF FIRST =0 THEN DO; /*INITIALIZE INDX ARRAY TO NULL */
  IF INDEX(RTP,'NOS')>0 THEN NOSSW='1'B;
  IF INDEX(RTP,'NOS')>0 THEN NOSSW='1'B;
  IF INDEX(RTP,'NEWSEC')>0 THEN SECSW='1'B;
     INDX=NULL;
     FIRST=1;
    END;
  REC.NUM=REC.NUM+1; /*TOTAL NUMBER OF RECORDS*/
   PUT SKIP(2) LIST(REFNO,REC.NUM);
  KNT=0;  /* NUMBER OF WORDS IN RECORD*/
LOOP1: DO I=1 TO 4;
     IF FILDS(I)='' THEN GO TO L1;
  IF FILDS(I)='FE' THEN DO; /*READ TERMS FROM HIT_LIST*/
       WRKSTR=PT->STR;
   PUT SKIP EDIT('    FE' :')(A);
          JJ=79;
       JK=1;
       GO TO LOOP2;
       END;
     JK=LOD-1;
LOOP2:  DO J=1 TO JK;
       IF FILDS(I)='FE' THEN GO TO LOOP3;

       IF TYPE(J)¬=FILDS(I) THEN GO TO LPND2;
  IF TYPE(J)='1     ' THEN LEN(J)=5;    /*LOOK AT 5 CHAR OF CODEN */
       PUT SKIP EDIT('   ',FILDS(I),':')(A,A,A);
       JJ=LEN(J);
  IF JJ> 999 THEN JJ = 999 ;
    WRKSTR=SUBSTR(STRNG,STRT(J),JJ); /*TRUNCATE AFTER 999 CHARS*/
LOOP3:  DO K=1 TO JJ WHILE(JJ>K); /*EXAMINE WRKSTR CHAR BY CHAR */
   IF SUBSTR(WRKSTR,K,3)='     ' THEN DO; /* SKIP OVER BLANKS*/
          K=K + JJ;
          GO TO LPND3;
          END;
  IF WRKST(K)='S' THEN GO TO HERE;
  IF WRKST(K)<'A' THEN GO TO LPND3; /*SKIP OVER NONALPHABETICS*/
       IF WRKST(K)>'Z' THEN GO TO LPND3;
HERE: DO KK=K + 1 TO JJ WHILE (WRKST(KK)¬=' ');
  END;   /*LOOK FOR END OF TERM*/
  IHK=KK;
  DO KK=IHK TO K BY -1 WHILE(WRKST(KK)<'A');  END;
  KK=KK-K+1; /*KK IS LENGTH OF TERM*/
  IF NOSSW THEN IF WRKST(K+KK-1)='S' THEN DO; /*REMOVE FINAL S */
   WRKST(K+KK-1)=' ';
   KK=KK-1;
  END;
     IF KK<3 THEN DO; /*SKIP OVER TERMS OF LENTH LESS THAN 3 */
         K=K + KK;
         GO TO LPND3;
         END;
```

```
        WORD=SUBSTR(WRKSTR,K,KK)
                          /* CH CK FOR STOPWORDS                    */
   NDX=0;
   IF KK=3 THEN NDX=INDEX(BADWRD3 WORD);
   IF KK=4 THEN NDX=INDEX(BADWRD4 WORD);
   IF KK=5 THEN NDX=INDEX(BADWRD5.WORD);
   IF KK=7 THEN NDX=INDEX(BADWRD7.WORD);
   IF KK=9 THEN NDX=INDEX(BADWRD9.WORD);
   IF NDX>0 THEN DO; /*WORD ON STOP LIST, SKIP OVER IT */
         K=K + KK;
         GO TO LPND3;
         END;
   IF INDEX(WORD,'$')>0 THEN DO;
   IF NO$SW THEN DO;
     K=K+KK;
     GO TO LPND3;
     END;
   ELSE IF SECSW THEN DO;
     IF SECON THEN DO;
       SECON='0'B;
     K=K-7;
       KK=6;
       WORD=SUBSTR(WORD,1,6);
       END;
     ELSE SECON='1'B;
     END;
   END;
         PUT EDIT(WORD)(X(1),A(KK));
   KNT=KNT+1; /* NUMBER OF WORDS IN RECORD*/
         II=INDEX(ALPH,SUBSTR(WORD,1,1)) + 27 - INDEX(ALPH,
         SUBSTR(WORD,2,1)); /*HASHING FUNCTION */
    IF INDX(II)=NULL THEN DO;/* FIRST TERM WITH THIS HASH CODE */
    ALLOCATE LTERM; /* ALLOCATE RECORD FOR THIS TERM */
    CNT=1; /* NUMBER OF OCCURANCES FOR THIS TERM */
         INDX(II)=P1;
         TERM=WORD;
         NUMWRDS=NUMWRDS + 1;
            RECN=RECNUM;
    NO=0; /* INDICATES TERM IS SINGULAR */
         NEXT=NULL;
         K=K + KK;
         LIST(KNT)=NO;
    PUT EDIT('(',NO,')')(A,F(3),A);
         GO TO LPND3;
         END;
   Q=INDX(II); /*HASH CODE PREVIOUSLY FOUND */
   IF Q->TERM>WORD THEN DO; /*TERM NOT PREVIOUSLY FOUND, SINCE
                               LIST IS IN ASCENDING ORDER */
   ALLOCATE LTERM /*ALLOCATE RECORD FOR THIS TERM */ ;
         CNT=1;
         TERM=WORD;
   INDX(II)=P1; /* PLACE TERM IN FRONT PF LIST */
         NUMWRDS=NUMWRDS + 1;
       NO=0;
            RECN=RECNUM;
         LIST(KNT)=NO;
```

```
       NEXT=Q; /*LINK TO NEXT RECORD IN LIST */
            K=K + KK;
        PUT EDIT('(',NO,')')(A,F(3),A);
             GO TO LPND3;
             END;
          Q=INDX(II);
 L9:      IF Q->TERM=WORD THEN DO;
             IF PH1 THEN IF RECNUM¬=Q->RECN THEN DO;
                Q->CNT=Q->CNT+1;
    IF Q->NO=0 THEN DO; /*WORD PREVIOUSLY FOUND, SHOULD BE MARKED
                      AS NONSINGULAR */
          NW=NW+1;
          Q->NO=NW;
              END;
       END;
            LIST(KNT)=Q->NO;
            K=K + KK;
        PUT EDIT('(',Q->NO,')')(A,F(3),A);
            GO TO LPND3;
            END;
       IF Q->TERM<WORD THEN DO; /*WORD MIGHT EXIST FURTHER ALONG THE
                      LIST*/
        IF Q->NEXT=NULL THEN DO;/*AT END OF LIST, SO WORD DID NOT
                      OCCUR PREVIOUSLY */
       ALLOCATE LTERM; /*ALLOCATE RECORD FOR THIS TERM */
            CNT=1;
       Q->NEXT=P1; /*PUT TERM AT END OF LIST */
              TERM=WORD;
              NUMWRDS=NUMWRDS + 1;
          NO=0;
              RECN=RECNUM;
              NEXT=NULL;
              LIST(KNT)=NO;
            K=K + KK;
        PUT EDIT('(',NO,')')(A,F(3),A);
              GO TO LPND3;
              END;
        LAST=Q; /* CONTINUE LOOKING DOWN LIST FOR TERM */
            Q=Q->NEXT;
            GO TO L9;
            END;
       ALLOCATE LTERM; /*TERM NOT FOUND, PLACE IN PROPER LOCATION */
            CNT=1;
          TERM=WORD;
       LAST->NEXT=P1; /*LINK TO NEXT TERM */
       NEXT=Q; /*LINK TO PREVIOUS TERM */
          NUMWRDS=NUMWRDS + 1;
          NO=0;
              RECN=RECNUM;
          LIST(KNT)=NO;
        PUT EDIT('(',NO,')')(A,F(3),A);
            K=K + KK;
 LPND3:   END LOOP3;
 LPND2:  END LOOP2;
 LPND1: END LOOP1;
 L1:
```

DM0700E7

```
      IF PH1 THEN RETURN; /*ONLY PRI T ON SECOND PASS.*/
       PUT FILE(PUNCH) EDIT(REC.NUM,ONE,KNT)
            (F(3),F(3),F(3));
      DO K=1 TO KNT; /*ONLY EXECUTED ON SECOND PAS SO NON-SINGULAR
                       TERMS CAN BE DISTIGUISHED. SINGULAR TERMS WILL
                       SHOW UP AS ZE OS IN THE LIST */
        PUT FILE(PUNCH) EDIT(REC.LI T(K))(F(3));
       END;
      PUT FILE(PUNCH) SKIP;
      RETURN;
   PRINTR: /* ONLY EXECUTED LAST TIM  TERMER IS CALLED */
      PUT PAGE;
      PUT FILE(PUNCH) EDIT(0,0,0)(F 3),F(3),F(3));
      DO I=2 TO 52;
       IF INDX(I)=NULL THEN GO TO L ;
       Q=INDX(I);
       DO WHILE (Q¬=NULL);
             PUT SKIP EDIT(Q->NO,Q-.TERM,Q->CNT)(F(3),X(2),A,
               X(2),F(3));
 PUT FILE(PUNCH) SKIP EDIT(Q->NO,Q->TERM)(F(3),A(20));
     /* Q->NO WILL BE 0 FOR SINGULA  TERMS AND A POSITIVE INTEGER
        FOR NON-SINGULAR TERMS */
         Q=Q->NEXT;
         END;
   LP:
      END;
      PUT SKIP(3) EDIT('TOTAL TERMS: ',NUMWRDS)(A,F(5));
      PUT SKIP(2) EDIT('NON-SINGULAR TERMS: ',NW)(A,F(5));
     RETURN;
     END TERMER;
```

SINORM: PROCEDURE OPTIONS (MAIN);

```
SINORM: PROCEDURE OPTIONS (MAIN);
/* FOR CLUSTERING. C6345, WE HAVE ON TAPE (IS1690,SRP.CASEC.50Z,FI
   THE CA VOLUME 65 SECTION 1 $ 2 INVERTED ON CA SECTION NUMBER.
   RECORDS LOOK LIKE:
        TERMA--SECTION1(FREQ),SECTION2(FREQ) ... SECTIONI(FREQ)
   IN THIS PROGRAM WE NORMALIZE THE FREQUENCY OF TH TERM BY SECTIO
   ACCORDING TO THE MAX TERMS IN ANY GIVEN SECTION (8088 IN THIS C
   THE RECORDS WRITTEN TO TAPE LOOK LIKE:
        TERMA--SECTION1(NORM FREQ),SECTION2(NORM FREQ),....,
          SECTIONI(NORM FREQ)
                                                              *SRP*/
DECLARE 1 OLDWRD BASED (PTR),
          2 NPOST FIXED BIN(15),
          2 WORD CHAR (20),
          2 FREQ FIXED BIN (15),
          2 MAX(100) DEC FIXED (6,2),
          2 POST(L REFER (OLDWRD.NPOST)) CHAR (6),
        TRMFRQ(80) DEC FIXED (6,2),
        J,K BIN FIXED (15) INIT (0),
        CASEC PICTURE '99.',
        UNQWRD FILE RECORD SEQUENTIAL INPUT,
        OUT FILE RECORD SEQUENTIAL OUTPUT;
        ON ENDFILE(UNQWRD) GO TO DONE;
                                   /* READ IN TOTAL FREQUENCY OF TERM
                                   /* FOR EACH SECTION.  WANT TO
                                   /* NORMALIZE BY MAX # TERMS THAT
                                   /* APPEAR IN ANY SECTION
        DO I=1 TO 80;
          GET LIST(TRMFRQ(I));
          TRMFRQ(I)=8088/TRMFRQ(I);
          END;

                                   /* PRINT OUT TABLE OF NORMALIZED
                                   /* FACTORS     */
        PUT SKIP EDIT('CASEC#','NORM FACTOR','CASEC#','NORM FACTOR
          (A(10),A,COL(50),A(10),A);
        DO I=1 TO 40;
        PUT SKIP EDIT(I,TRMFRQ(I),I+40,TRMFRQ(I+40))
          (F(6),COL(11),F(6,2),COL(50),F(6),COL(66),F(6,2));
          END;

                                   /* READ A BLOCK CONTAINING A TERM
                                   /* AND UP TO 100 POSTINGS
        J=0;
READ:   READ FILE (UNQWRD) SET (PTR);
        J=J+1;                     /* COUNT # BLOCKS

INCRM:  DO K=1 TO OLDWRD.NPOST;  /* LOOK AT ALL POSTINGS FOR TERM
        CASEC=SUBSTR(OLDWRD.POST(K),1,3);
        MAX(K)=MAX(K)*TRMFRQ(CASEC);
          END;
        WRITE FILE (OUT) FROM (OLDWRD);
        GO TO READ;
DONE:   PUT SKIP EDIT(J,' BLOCKS OF RECORDS PROCESSED')(F(6),A);
        END SINORM;
```

```
S2SUPER:PROCEDURE OPTIONS(MAIN);
/* FOR STEP2 OF THE CLUSTERING TERM MAPPING EXPERIMENTS, WE WANT TO FOR
   EACH TERM, ADD UP ALL OCCURANCES OF THAT TERM IN ALL CA SECTIONS
   NORMALIZED.  THEN FIND THE BIGGEST SECTION (CONTAINING MOST
   OCCURANCES) AND COMPUTE:
        BIGGEST/TOTAL=F1
   FOUR ARRAYS OF 100 POSITIONS ARE DECLARED:
        1ST VECTOR IS FOR 1ST BIGGEST PEAK >MIN FREQUENCY OF OCCURANCES
        2ND VECTOR IS FOR 2ND BIGGEST PEAK >MIN FREQUENCY OF OCCURANCES
        3RD VECTOR IS FOR 1ST BIGGEST PEAK <MIN FREQUENCY OF OCCURANCES
        4TH VECTOR IS FOR 2ND BIGGEST PEAK <MIN FREQUENCY OF OCCURANCES
   THE APPROPIEATE SPOT IN ARRAY IS INCREMENTED BY ONE FOR EACH ENTRY.
                                                                *SRP*/
DECLARE 1 WRD BASED (PTR),
          2 NPOST FIXED BIN (15),
          2 WORD CHAR (20),
          2 FREQ FIXED BIN (15),
          2 MAX(100) DEC FIXED (6,2),
          2 POST(L REFER (NPOST)) CHAR(6),
        LASTWORD CHAR (20) INIT ('                    '),
        (LCASEC,CASEC) PICTURE '999',
        (FIRSTSEC,SECSEC) PICTURE '999',
        (MINFREQ,NUMBLK) DEC FIXED (6),
        SUPER(5) DEC FIXED (6,2),
        (FIRSTPAK,SECPAK,WRDCNT) DEC FIXED (6,2),
        (F1,F2) DEC FIXED (6),
        SW BIT (1) INIT ('0'B),
        M DEC FIXED (3),          /* COUNTER FOR SUPER SECTIONS */
        IN FILE RECORD SEQUENTIAL INPUT;
DECLARE (ARRAY1(100),ARRAY2(100),ARRAY3(100),ARRAY4(100)) DEC FIXED (6)
;
       DECLARE UNDERLINE CHAR (66);

       ON ENDFILE(IN) GO TO DONE;
       OPEN FILE(SYSPRINT) STREAM OUTPUT PRINT PAGESIZE(56)
         LINESIZE(132);
```

```
      ON ENDPAGE (SYSPRINT) BEGIN;
       IF ¬SW THEN DO;
         PUT PAGE;
         PUT EDIT('1ST BIGGEST','2ND BIGGEST','1ST BIGGEST',
         '2ND BIGGEST')(COL(3,),A,COL(49),A,COL(95),A,COL(114),A);
       PUT SKIP;
       PUT EDIT('WORD','TOTAL','SUPERSEC','%','SUPERSEC','%')
         (X(3),A(21),A(8),A(11),X(1),A(1),X(6),A(11),X(2),A(1));
       PUT EDIT('WORD','TOTAL','SUPERSEC','%','SUPERSEC','%')
         (X(3),A(21),A(8),A(11),X(1),A(1),X(6),A(11),X(2),A(1));
       PUT SKIP:
       PUT EDIT(UNDERLINE,UNDERLINE)(X(3),A(66),X(3),A(60));
        PUT SKIP;
         END;
      END;



      GET LIST(NUMBLK,MINFREQ);
      PUT SKIP EDIT(NUMBLK,' BLOCKS TO BE PROCESSED')(F(6),A);
      PUT SKIP EDIT(MINFREQ,' IS MINIMUM FREQUENCY')(F(6),A);

      WRDCNT=0;  SUPER=0;
      ARRAY1=0;  ARRAY2=0;  ARRAY3=0:  ARRAY4=0;
      FIRSTPAK=0;  SECPAK=0;
      FIRSTSEC=0;   SECSEC=0:
      I=0:
      UNDERLINE='_______________________________________________________
_______________';
      SIGNAL ENDPAGE(SYSPRINT);
READ: READ FILE(IN) SET (PTR);
      I=I+1;
      IF I>NUMBLK THEN GO TO DONE;  /* PROCESSED ENOUGH?             */

INCRM: DO K=1 TO NPOST;         /* READ ALL POSTINGS IN BLOCK      */
        CASEC=SUBSTR(POST(K),1,3);  /* EXTRACT ONLY SEC NUMBER     */

      IF CASEC<=20 THEN M=1:
        ELSE IF CASEC<=34 THEN M=2;
        ELSE IF CASEC<=46 THEN M=3;
        ELSE IF CASEC<=64 THEN M=4;
        ELSE IF CASEC<=80 THEN M=5;
      SUPER(M)=SUPER(M)+MAX(K);
        WRDCNT=WRDCNT+MAX(K);   /* COUNT NUMBER OF WORDS FOR TERM  */

      IF SUPER(M)>FIRSTPAK THEN DO;
      IF M¬=FIRSTSEC THEN DO:
      SECPAK=FIRSTPAK;
      SECSEC=FIRSTSEC;
        END;
      FIRSTPAK=SUPER(M);
      FIRSTSEC=M;
        END;
```

```
EINCRM:  END INCRM;

         LASTWORD=WORD;
         READ FILE(IN) SET (PTR); /* READ NEXT BLOCK          */
         I=I+1;                   /* COUNT NUMBER OF BLOCKS              */
         IF I>NUMBLK THEN GO TO DONE;  /* PROCESSED ENOUGH?

                                  /* IF THIS BLOCK IS OF SAME WORD AS */
                                  /* LAST CONTINUE INCREMENTING       */
         IF LASTWORD=WORD THEN GO TO INCRM;

                                  /* THE FOLLOWING DOES ACTUAL         */
                                  /* ADDITION INTO ARRAYS   */
         F1=FIRSTPAK/WRDCNT*100;  /* COMPUTE 1ST PEAK FOR THIS TERM   */
         F2=SECPAK/WRDCNT*100;    /* COMPUTE 2ND PEAK FOR THIS TERM   */

                                  /* THE ARRAYS TO WHICH RESULT IS    */
                                  /* ASSIGNED DEPENDS WHETHER WORD IS */
                                  /* LESS THAN OR GREATER THAN        */
                                  /* MINIMUM FREQUENCY                */
         IF WRDCNT>=MINFREQ THEN DO;
           ARRAY1(F1)=ARRAY1(F1)+1;
           ARRAY2(F2)=ARRAY2(F2)+1;
           END;
         ELSE DO;
           ARRAY3(F1)=ARRAY3(F1)+1;
           ARRAY4(F2)=ARRAY4(F2)+1;
           END;
         PUT EDIT(LASTWORD)(X(3),A(21));
         PUT EDIT(WRDCNT,FIRSTSEC,F1,SECSEC,F2,' ')
           (F(6,2),X(4),F(3),X(7),F(3),X(7),F(3),X(6),F(3),A(1));
         FIRSTSEC=0;  SECSEC=0;
         FIRSTPAK=0;  SECPAK=0;
         WRDCNT=0;    SUPER=0;
         GO TO INCRM;
DONE:    PUT PAGE EDIT('1ST PEAK> ',MINFREQ,'2ND PEAK> ',MINFREQ,
           '1ST PEAK< ',MINFREQ,'2ND PEAK < ',MINFREQ)(A,F(6),X(7));
         SW='1'B;
         DO J=1 TO 100;
           PUT SKIP EDIT(ARRAY1(J),ARRAY2(J),ARRAY3(J),ARRAY4(J))
           (X(5),F(6),X(10));
           END;
         PUT SKIP EDIT(I,' BLOCKS READ')(F(6),A);
         END S2SUPER;
```

```
S2SEC:  PROCEDURE OPTIONS (MAIN);
/* FOR STEP2 OF THE CLUSTERING TERM MAPPING EXPERIMENTS, WE WANT TO FOR
   EACH TERM, ADD UP ALL OCCURANCES OF THAT TERM IN ALL CA SECTIONS
   NORMALIZED.  THEN FIND THE BIGGEST SECTION (CONTAINING MOST
   OCCURANCES) AND COMPUTE:
         BIGGEST/TOTAL=F1
   FOUR ARRAYS OF 100 POSITIONS ARE DECLARED:
         1ST VECTOR IS FOR 1ST BIGGEST PEAK >MIN FREQUENCY OF OCCURANCES
         2ND VECTOR IS FOR 2ND BIGGEST PEAK >MIN FREQUENCY OF OCCURANCES
         3RD VECTOR IS FOR 1ST BIGGEST PEAK <MIN FREQUENCY OF OCCURANCES
         4TH VECTOR IS FOR 2ND BIGGEST PEAK <MIN FREQUENCY OF OCCURANCES
   THE APPROPIEATE SPOT IN ARRAY IS INCREMENTED BY ONE FOR EACH ENTRY.
                                                                  *SRP*/
DECLARE 1 WRD BASED (PTR),
           2 NPOST FIXED BIN (15),
           2 WORD CHAR (20),
           2 FREQ FIXED BIN (15),
           2 MAX(100) DEC FIXED (6,2),
           2 POST(L REFER (NPOST)) CHAR(6),
        LASTWORD CHAR (20) INIT ('                    '),
        (LCASEC,CASEC) PICTURE '999',
        (FIRSTSEC,SECSEC) PICTURE '999',
        (MINFREQ,NUMBLK) DEC FIXED (6),
        SEC(80) DEC FIXED (6,2),
        (FIRSTPAK,SECPAK,WRDCNT) DEC FIXED (6,2),
        (F1,F2) DEC FIXED (6),
        SW BIT (1) INIT ('0'B),
        IN FILE RECORD SEQUENTIAL INPUT;
DECLARE (ARRAY1(100),ARRAY2(100),ARRAY3(100),ARRAY4(100)) DEC FIXED (6)
;
        DECLARE UNDERLINE CHAR (66);

        ON ENDFILE(IN) GO TO DONE;
        OPEN FILE(SYSPRINT) STREAM OUTPUT PRINT PAGESIZE(56)
          LINESIZE(132);





        ON ENDPAGE (SYSPRINT) BEGIN;
```

```
IF ¬SW THEN DO;
  PUT PAGE;
  PUT EDIT('1ST BIGGEST','2ND BIGGEST','1ST BIGGEST',
  '2ND BIGGEST')(COL(27),A,COL(46),A,COL(93),A,COL(112),A);
PUT SKIP;
PUT EDIT('WORD','TOTAL','CASEC','%','CASEC','%')
  (X(3),A(21),A(8),X(1),A(11),X(1),A(1),X(6),A(11),X(2),A(1));
PUT EDIT('WORD','TOTAL','CASEC','%','CASEC','%')
  (X(3),A(21),A(8),X(1),A(11),X(1),A(1),X(6),A(11),X(2),A(1));
PUT SKIP;
PUT EDIT(UNDERLINE,UNDERLINE)(X(3),A(66),X(3),A(66));
PUT SKIP;
  END;
END;




GET LIST(NUMBLK,MINFREQ);
PUT SKIP EDIT(NUMBLK,' BLOCKS TO BE PROCESSED')(F(6),A);
PUT SKIP EDIT(MINFREQ,' IS MINIMUM FREQUENCY')(F(6),A);

LCASEC=000;
WRDCNT=0;    SEC=0;
ARRAY1=0;  ARRAY2=0;  ARRAY3=0:  ARRAY4=0;

FIRSTPAK=0;  SECPAK=0;
FIRSTSEC=0;   SECSEC=0;
I=0;
UNDERLINE='__________________________________________________________
_______________';
SIGNAL ENDPAGE(SYSPRINT);
READ:    READ FILE(IN) SET (PTR);
         I=I+1;
         IF I>NUMBLK THEN GO TO DONE;   /* PROCESSED ENOUGH?              */

INCRM:   DO K=1 TO NPOST;          /* READ ALL POSTINGS IN BLOCK        */
           CASEC=SUBSTR(POST(K),1,3);  /* EXTRACT ONLY SEC NUMBER       */

                                   /* IS THIS SECTION IS THE SAME AS    */
                                   /* THE LAST SECTION; ADD TOGETHER    */
           IF CASEC=LCASEC THEN SEC(CASEC)=SEC(CASEC)+MAX(K);

                                   /* OTHERWISE ASSIGN FREQUENCY        */
           ELSE SEC(CASEC)=MAX(K);
           WRDCNT=WRDCNT+MAX(K);  /* COUNT NUMBER OF WORDS FOR TERM    */

         IF SEC(CASEC)>FIRSTPAK THEN DO;
         IF CASEC¬=FIRSTPAK THEN DO;
           SECPAK=FIRSTPAK;
           SECSEC=FIRSTSEC;
           END;
           FIRSTSEC=CASEC;
           FIRSTPAK=SEC(CASEC);
           END;
```

```
        LCASEC=CASEC;
 EINCRM:   END INCRM;

        LASTWORD=WORD;
        READ FILE(IN) SET (PTR); /* READ NEXT BLOCK               */
        I=I+1;                   /* COUNT NUMBER OF BLOCKS              */
        IF I>NUMBLK THEN GO TO DONE;  /* PROCESSED ENOUGH?

                                 /* IF THIS BLOCK IS OF SAME WORD AS */
                                 /* LAST CONTINUE INCREMENTING       */
        IF LASTWORD=WORD THEN GO TO INCRM;

                                 /* THE FOLLOWING DOES ACTUAL        */

                                 /* ADDITION INTO ARRAYS   */
        F1=FIRSTPAK/WRDCNT*100;  /* COMPUTE 1ST PEAK FOR THIS TERM   */
        F2=SECPAK/WRDCNT*100;    /* COMPUTE 2ND PEAK FOR THIS TERM   */

                                 /* THE ARRAYS TO WHICH RESULT IS    */
                                 /* ASSIGNED DEPENDS WHETHER WORD IS */
                                 /* LESS THAN OR GREATER THAN        */
                                 /* MINIMUM FREQUENCY                */
        IF WRDCNT>=MINFREQ THEN DO;
          ARRAY1(F1)=ARRAY1(F1)+1;
          ARRAY2(F2)=ARRAY2(F2)+1;
          END;
        ELSE DO;
          ARRAY3(F1)=ARRAY3(F1)+1;
          ARRAY4(F2)=ARRAY4(F2)+1;
          END;
        PUT EDIT(LASTWORD)(X(3),A(21));
        PUT EDIT(WRDCNT,FIRSTSEC,F1,SECSEC,F2,' ')
          (F(6,2),X(4),F(3),X(5),F(3),X(7),F(3),X(6),F(3),A(1));
        FIRSTSEC=0;  SECSEC=0;
        FIRSTPAK=0;  SECPAK=0;
        WRDCNT=0;   SEC=0;
        GO TO INCRM;
 DONE:  PUT PAGE EDIT('1ST PEAK> ',MINFREQ,'2ND PEAK> ',MINFREQ,
          '1ST PEAK< ',MINFREQ,'2ND PEAK < ',MINFREQ)(A,F(6),X(7));
        SW='1'B;
        DO J=1 TO 100;
          PUT SKIP EDIT(ARRAY1(J),ARRAY2(J),ARRAY3(J),ARRAY4(J))
          (X(5),F(6),X(10));
          END;
        PUT SKIP EDIT(I,' BLOCKS READ')(F(6),A);
        END S2SEC;
```

```
                                          /* L ST UPDATE:    760824;        SRP*/
INVERT: PROC REORDER OPTIONS(MAIN);
                                           /* P OGRAM: INVERT  MODULE NO: 43  */
                                           /* T IS PROGRAM IS THE FIRST PHASE */
                                           /* O  THE INVERSION PROCESS; IT    */
                                           /* P LLS OUT EVERY WORD IN THE     */
                                           /* T TLE AND KEYWORDS ELEMENTS OF  */
                                           /* I-TRI-FORMAT RECORDS AND PUTS   */
                                          /* EA H WORD, WITH THE SPECIFIED   */
                                          /* PO TING ATTACHED, TO A FILE FOR */
                                          /* SO TING.  THIS IS A MODIFICATION */
                                          /* OF DM069043 FOR CLUSTERING EXPER.*/

     /* CHANGED FOR NEW FORMAT; S.E.P. JULY 1974                    */
     DECLARE
ONSOURCE BUILTIN,
CHK CHAR(3) STATIC INIT('   '),
STOP CHAR(24) STATIC INIT('OF AND THE IN ON FOR BY'),
     (AVERAGE,DNUMREC,DKOUNTR) DE  FIXED(10,2),
     1 DRTRY BASED(PLSRP),
          2 UNO CHAR(10),
          2 ABSTNUM CHAR(11),
          2 PAD CHAR(1),
          2 DAT FIXED BIN(15),
          2 LOD FIXED BIN(15),
          2 LOP FIXED BIN(15),
          2 DIR(1),
               3 TYPE CHAR(4),
               3 ST FIXED BIN(15),
               3 LN FIXED BIN(15),
    (TYP,FT1,FT2) CHAR(4) STATIC,
     (NUMBER,NUMREC,KOUNTR,HINUM, RIV) BIN FIXED(31),
                                          /* NOTE CAREFULLY THE FOLLOWING    */
                                          /* OVERLAYS, THEY ARE VITAL IN     */
                                          /* UNDERSTANDING THE DATA MOVEMENT */
                                          /* A D CHARACTER EXAMINATION       */
                                          /* R UTINES                        */
LIST1 CHAR(1000) STATIC INIT(' '),
ARR1(1000) CHAR(1) DEF LIST1,
LIST2 CHAR(255) BASED(LPTR),
```

DM070043

```
ARR2(1) CHAR(1) BASED(SPTR),
(NDX01,NDX02,NDX03,NDX04) FIXED BIN(16) STATIC INIT(0),
STPT FIXED BIN(16) STATIC INIT(0),
(LPTR,SPTR,QPTR) PTR,
(LCIT,LCIT2) FIXED BIN(31) STATIC INIT(0),
WORDSP CHAR(40) STATIC INIT(' '),
WORDX CHAR(20) DEF WORDSP,
        1 WRD,
          2 WORD CHAR(20),
          2 POSTING CHAR (6),
        PFIELD CHAR (4),
        FMTFILE FILE RECORD SEQUENTI L INPUT,
        WORDS FILE RECORD SEQUENTIAL OUTPUT;
DECLARE HOLD CHAR (20),            /* IN UT VARIABLE FOR PRINTING LIST */
        PTSW BIT(1) INIT ('1'B),
        I BIN FIXED (31),
        TRMFRQ(80) BIN FIXED (31), /* RRAY FOR CA SEC# TERM FREQ    */
        CASEC PICTURE '999',
        PLSSTR CHAR(1000) BASED (SPTR);
DECLARE DASH CHAR (19) INIT ('-------------------');
        KOUNTR,NUMREC,TRIV=0;

ON ERROR BEGIN;
 PUT SKIP(6) EDIT('ERROR AT ',ORTRY.ABSTNUM)(A,A);
 GOTO ENDPGM;
END;
ON CONVERSION ONSOURCE=0;
        TRMFRQ=0;                  /* INITIALIZE ARRAY OF TERM FREQ.    */
                                   /* READ LIMIT ON CITATIONS TO BE     */
                                   /* PROCESSED                         */
                                   /* A D FIELDS TO INVERT */
         GET EDIT(NUMBER,FT1,FT2)(F(6),A(4),A(4));
  PUT SKIP EDIT('LIMIT:',NUMBER,' CITATIONS. FIELDS: ',
   FT1,FT2)(A,F(7),A,A,X(2),A);
                                   /* READ FIELD TO BE USED FOR POSTING*/
        GET SKIP EDIT(PFIELD)(A(4));
        PUT SKIP EDIT('FIELD USED FOR POSTING: ',PFIELD)(A);
        PUT SKIP;
        GET SKIP EDIT(HOLD)(A(7));
        PUT SKIP EDIT(HOLD)(A);
        IF HOLD='NOPRINT' THEN PTSW='0'B;
        ON ENDFILE(FMTFILE) GO TO ENDPGM;
        ON RECORD(FMTFILE) BEGIN; END;
                                   /*                                   */
START:
         READ FILE(FMTFILE) SET(PLSRP);
         SPTR=ADDR(TYPE(LOD+1));




        KOUNTR=KOUNTR+1;
LEN=0;
NDX02=0;
                                   /* FIRST LOOP LOOKS FOR TITLE        */
```

```
                                          /* FIELD; IF IT FINDS THE KEYWORDS */
                                          /* FIRST, IT REMEMBERS THAT FOR    */
                                          /* L TER                           */
    LOOP01:
         DO NDX01= 1 TO LOD-1;

              TYP=TYPE(NDX01);
              IF TYP=FT1 THEN GOTO FOU D1;
              IF TYP=FT2 THEN NDX02=NDX01;
          END LOOP01;
        GO TO LOOP02;
 FOUND1:
         LEN=LN(NDX01);
                                          /* T E NEXT SECTION MOVES THE TITLE*/
                                          /* T  A WORK AREA TO EXAMINE IT,   */
                                          /* I  A REASONABLY EFFICIENT MANNER*/
 LPTR=ADDR(ARR1(1));
         QPTR=ADDR(ARR2(ST(NDX01)));
 IF LEN<=85 THEN SUBSTR(LPTR->LIST2,1,85)=SUBSTR(QPTR->LIST2,1,85);
 ELSE IF LEN<=140 THEN
  SUBSTR(LPTR->LIST2,1,140)=SUBSTR(QPTR->LIST2,1,140);
 ELSE DO;
  LPTR->LIST2=QPTR->LIST2;
  IF LEN>255 THEN DO;
  LPTR=ADDR(ARR1(256));
    QPTR=ADDR(ARR2(ST(NDX01)+255));
   LPTR->LIST2=QPTR->LIST2;
                                          /* I  LONGER THAN 510 CHARACTERS   */
                                          /* T E FIELD IS TRUNCATED          */
   LCIT=LCIT+1;
   IF LEN>510 THEN DO;
     LCIT2=LCIT2 + 1;
     LEN=510;
     END;
  END;
 END;
                                          /*                                 */
                                          /* IF THE KEYWORD FIELD WASN'T     */
                                          /* FOUND BEFORE, IT IS NOW SOUGHT  */
 LOOP02:
 IF NDX02>0 THEN GOTO FOUND2;
        DO NDX02=NDX01 TO LOD-1;
             TYP=TYPE(NDX02);
             IF TYP=FT2 THEN GOTO FOU D2;
 END;
       GO TO ON01;
                                          /* T E KEYWORDS ARE NOW MOVED      */
                                          /* SIMILARLY TO WORK AREA FOLLOWING*/
                                          /* T E TITLE                       */
 FOUND2:
 ARR1(LEN+1)=' ';
        LEN2=LN(NDX02);
 LPTR=ADDR(ARR1(LEN+2));
        QPTR=ADDR(ARR2(ST(NDX02)));
 IF LEN2<=65 THEN SUBSTR(LPTR->LIST2,1,65)=SUBSTR(QPTR->LIST2,1,65);
 ELSE IF LEN2<=110 THEN
```

```
 SUBSTR(LPTR->LIST2,1,110)=SUBSTR(QPTR->LIST2,1,110);
ELSE DO;
 LPTR->LIST2=QPTR->LIST2;
 IF LEN2>255 THEN DO;
  LPTR=ADDR(ARR1(LEN+257));
    QPTR=ADDR(ARR2(ST(NDX02)+255));
  LPTR->LIST2=QPTR->LIST2;
  LCIT=LCIT+1;
  IF LEN>510 THEN DO;
    LCIT2=LCIT2 + 1;
    LEN=510;
    END;
 END;
END;
LEN=LEN+LEN2+1;
                                          /*                                  */
                                          /* NOW THE WORDS MUST BE BROKEN OUT*/
                                          /* OF BOTH TITLE AND KEYS           */
ONO1:
IF LEN=0 THEN GOTO START;
        STPT=1;
LPTR=ADDR(ARR1(1));
LEN=LEN+1;
ARR1(LEN)=' ';
                                          /* LOOP AHEAD TO A NON-ALPHABETIC   */
                                          /* CHARACTER                        */
LOOP03: DO NDX03=1 TO LEN BY 1;
IF ARR1(NDX03)>='A' THEN GOTO FLOOP3;
                                          /* THE ELSE BLOCK CHECKS FOR AN     */
                                          /* ACCEPTABLE WORD, REJECTING IF    */
                                          /* ONE CHARACTER, BEGINS WITH A     */
                                          /* NUMBER, OR APPEARS IN THE QUICK  */
                                          /* STOP LIST                        */
        ELSE DO;
LEN2=NDX03-STPT;
IF LEN2<4 THEN DO;
IF LEN2<2 THEN GOTO NO;
 CHK=SUBSTR(LIST2,STPT,3);
IF SUBSTR(CHK,1,1)>'Z' THEN GOTO NO;
WORD=CHK;
IF INDEX(STOP,CHK)>0 THEN GOTO NO;
END;
ELSE DO;
IF ARR1(STPT)>'Z' THEN GOTO NO;
IF LEN2>=20 THEN WORD=SUBSTR(LIST2,STPT,20);
ELSE DO;
 WORDX=SUBSTR(LIST2,STPT,20);
 SUBSTR(WORDSP,LEN2+1,20)='                    ';
                                        /* EXTRACT POSTING                    */
          DO NDX01=1 TO LOD-1;
      IF TYPE(NDX01)=PFIELD & PFIELD='1    ' THEN
        POSTING=SUBSTR(PLSSTR,ST(NDX 1),6);
      IF TYPE(NDX01)=PFIELD & PFIELD='5    ' THEN
        POSTING=SUBSTR(PLSSTR,ST(NDX 1)+3,6);
        END;
 WORD=WORDX;
```

```
END;
END;
      IF PTSW THEN PUT EDIT(WORD,POSTING)(A(20),A(10));
      IF PFIELD='5    ' THEN DO;
        CASEC=SUBSTR(POSTING,1,3);
        TRMFRQ(CASEC)=TRMFRQ(CASEC)+ ;
        END;
ONO2:
                                    /* HE E WORD AND POSTING ARE          */
                                     /* W ITTEN                            */
          WRITE FILE(WORDS) FROM(WRD);
          NUMREC=NUMREC+1;
                                     /* SKIP HERE TO GO ON AFTER           */
                                     /* R JECTED TERM                      */
NO:       STPT=NDX03+1;
          END;
ELOOP3:   END LOOP03;
ENDCHK: IF KOUNTR<NUMBER THEN GO TO START;
                                     /*                                    */
                                     /* E D OF PROGRAM, PRINT STATISTICS*/
ENDPGM: CLOSE FILE(FMTFILE), FILE(WORDS);
        PUT EDIT('NUMBER OF CITATIONS PROCESSED:',KOUNTR)(PAGE,A(30),
          F(8));
PUT EDIT('NUMBER OF POSTINGS ',NUMREC)(SKIP(2),A,
          F(8));
PUT SKIP(2) EDIT('FULL LENGTH MOVE USE  ',LCIT,' TIMES.')(A,F(4),A);
PUT SKIP(1) EDIT('  TRUNCATION OCCURRE  ',LCIT2,' TIMES.')(A,F(4),A);
        DNUMREC=NUMREC;
        DKOUNTR=KOUNTR;
        AVERAGE=DNUMREC/DKOUNTR;
PUT EDIT('MEAN NUMBER OF POSTINGS PER CITATION ',
          AVERAGE)(SKIP(2),A(37),F(10, ));
                                    /* PR NT FREQUENCY OF TERMS FOR EACH*/
                                    /* CA SECTION #                        */
        IF PFIELD='5    ' THEN DO;
          PUT PAGE EDIT('CASEC#','TOTA  FREQ OF TERMS','CA SEC#',
            'TOTAL FREQ OF TERMS')(A(1 ),A,COL(50),A(10),A);
PUT SKIP EDIT(DASH,DASH,DASH,DASH)(A(1 ),A(19),COL(50),A(10),A(19));
          DO I=1 TO 40;
            PUT SKIP EDIT(I,TRMFRQ(I),I+40,TRMFRQ(I+40))
              (F(6),COL(11),F(6),COL(5 ),F(6),COL(66),F(6));
            END;
          END;
        END INVERT;
```

```
                                          /* LAST UPDATE: 760630 */
                                          /* LAST UPDATE: 751001 */
SQUEEZ: PROC REORDER OPTIONS(MAIN);
                                  /* PROGRAM: SQUEEZE  MODULE NO.: 44*/
                                  /* THIS PROGRAM READS THE  (SORTED)*/
                                  /* POSTINGS FROM THE INVERT PROGRAM*/
                                  /* AND CREATES BLOCKS OF UP TO 100.*/
                                  /* THIS IS A MODIFICATION OF       */
                                  /* D 069044 FOR EXPERIMENTS FOR    */
                                  /* CLUSTERING.  A POSTING CAN BE   */
                                  /* EITHER A CODEN(6 CHAR ) OR 6    */
                                  /* DIGITS OF THE CA SECTION #      */

DECLARE
      1 WRD BASED(WPTR),        /* WORD AND CODEN NOW SORTED         */
        2 WORD CHAR (20),
         2 POST CHAR (6),
      1 OLDWRD BASED (OWPTR),
         2 NPOST   FIXED BIN (15), /* NO. OF POSTINGS IN THIS BLOCK */
        2  OLDWORD CHAR (20),
         2 FREQ FIXED BIN(15),  /* NO. IN ALL BLOCKS THUS FAR      */
         2 MAX(100) DEC FIXED (6,2),
         2 POST(K REFER(NPOST)) CHAR (6),
        K FIXED BIN(15) STATIC INIT(0),
        (TDUP,DUP,COL,LIN) FIXED BIN(31) INIT(0),
        LPOST CHAR (6),
        (L0,L2,L3) FIXED BIN (31) STATIC INIT(0),
       (STOP(0:122),HOLD) CHAR (20), /* STOP LIST                    */

        THL CHAR(44) STATIC
          INIT('TERM                  NPOST   CITS  FREQ'),
        ITIM CHAR(44) STATIC INIT(' '),
        PIM(50) CHAR(132),
         (UPP,LOW,DIV,SAVER) BIN FIXED,
         (TOTAL,NUM,J,L,M) BIN FIXED(31),
         (DJ,DL,AVERAGE) DEC FIXED(10,2),
        (PTSW, ASW)  BIT(1) ALIGNED STATIC,
         WORDS FILE RECORD SEQUENTIAL INPUT,
         UNQWRD FILE RECORD SEQUENTIAL OUTPUT;
       OPEN FILE(WORDS), FILE(UNQWRD)
        OPEN FILE(SYSPRINT) PRINT LINESIZE(132) PAGESIZE(55);

       ON ENDFILE(WORDS) GO TO ENDPGM
       ON ENDFILE(SYSIN) GO TO CONTIN
                             /* BUILD A MAXIMUM-SIZE BLOCK TO   */
                             /* FILL LATER                      */
        K=100;
                                 /* CHECK WHETHER TO PRINT FREQUENCY */
                                 /* LIST BY READING CARD FROM SYSIN  */
ALLOCATE OLDWRD SET(OWPTR);
       GET EDIT(HOLD)(A(20));
        IF HOLD='NOPRINT' THEN PTSW='0'B;
        ELSE PTSW='1'B;
       MAX=1;
       LOW,I,ITRIV,NTRI=0;
        TDUP,DUP,LPOST,LIN=0;  COL=1;  ITIM=' ';
                             /* READ THE STOP LIST              */
```

```
 READ:    GET EDIT(HOLD)(SKIP(1),A(20)),
          STOP(I)=HOLD;
          I=I+1;
          GO TO READ;
MCONTIN:  SAVER=I;                      /* NUMBER OF STOP WORDS        */

          UPP=2;
                                    /* SET UPPER BOUND FOR BINARY     */   :
                                    /* SEARCH (POWER OF 2)            */   :
 LOOP00:  DO WHILE (UPP<SAVER);
            UPP=UPP*2;
            END LOOP00;
          ILIMIT=UPP;
          TOTAL,I,J,L,M=0;
 READ FILE(WORDS) SET(WPTR);
          OLDWRD.OLDWORD=WRD.WORD;
                                         /* THE NEXT BLOCK IS A NORMAL     */ :
                                         /* BINARY SEARCH OF THE STOP LIST */ :
                                         /* TO DETERMINE IF THE TERM LIES  */ :
                                         /* THEREIN                        */ :
          LOW=0;
          UPP=ILIMIT;
          DIV=UPP/2;
 COMPR:   IF DIV>SAVER THEN DO;
            UPP=DIV;
            GO TO COMPT;
            END;
          HOLD=STOP(DIV);
          IF OLDWORD<HOLD THEN DO;
            UPP=DIV;
            GO TO COMPT;
            END;
          IF OLDWORD>HOLD THEN DO;
            LOW=DIV;
            GO TO COMPT;
            END;
          GO TO ON00;
 COMPT:   DIV=(LOW+UPP)/2;
          IF DIV ¬= LOW THEN GO TO COMPR;
                         ELSE GO TO ON;
                              /* IF WOR  IS IN STOP LIST, SET    */     :
                              /* ITRIV=                          */     :
 ON00:    ITRIV=1;
          GO TO READER;

                              /* OTHERWISE SET ITRIV=0 AND SET UP*/     :
                              /* OUTPUT BLOCK                    */     :
 ON:      ITRIV=0;
          OLDWRD.POST(1)=WRD.POST;
 UNIQUE:  K,NUM=1;
          TDUP=TDUP+DUP;    DUP=0;
 LO=0;
 READER:
 READ FILE(WORDS) SET(WPTR);          /* RE D AN EXTRACTED WORD & POSTING */

                                      /* TH S PROCESSING INVOLVES WORDS   */
                                      /* WH CH HAVE BEEN ENTERED BEFORE   */
```

```
   IF WRD.WORD=OLDWRD.OLDWORD THEN DO;
       ASW='0'B;
     IF ITRIV=1 THEN GO TO READER; /* SKIP IF A STOP WORD                  */

      ELSE DO;
                                  /* CHECK IF WORD APPEARED IN SAME      */
                                  /* POSTING   */
       IF WRD.POST=LPOST THEN DO;
         DUP=DUP+1;
         MAX(K)=MAX(K)+1;    END;

                                  /* IF BLOCK IS FULL, WRITE IT           */
       ELSE DO;
         IF K=100 THEN DO;
            IF L0=0 THEN DO; L2=L2+1; L0=1; END;
             I=1;
            GO TO WRITER;
            END;
                              /* THEN, OR OTHERWISE, SET POSTING */
                              /* IN BLOCK                         */
        K=K+1;
         NUM=NUM+1;               /* COUNTS NO. POSTINGS PER WORD         */
         IF NUM>M THEN M=NUM;     /* HIGHEST NO. POSTINGS PER WORD        */

          OLDWRD.POST(K)=WRD.POST;
         LPOST=WRD.POST;
          J=J+1;
       END;
       GOTO READER;
     END;


   END;
   ELSE ASW='1'B;

                              /*                                  */
WRITER: IF ITRIV=1 THEN DO;
        NTRI=NTRI+1;
          GO TO ON01;
          END;
 IF L0=1 THEN DO;
  L=L+1;
 L3=L3+K;
 END;
 NPOST=K;
 OLDWRD.FREQ=NUM;
        WRITE FILE(UNQWRQ) FROM(OLDWRD);
  IF ASW THEN
    IF PTSW THEN CALL PRNT;
        MAX=1;
 K=1;
        J=J+1;
                                  /* THIS GROUP IS EXECUTED AFTER FULL*/
                                  /* BLOCK IS WRITTEN.  SET FLAG BACK */
                                  /* TO 0 & POSTING BECOMES FIRST     */
                                  /* POSTING OF NEW BLOCK             */
        IF I=1 THEN DO;
```

```
          I=0;
          OLDWRD.POST(1)=WRD.POST;
          LPOST=WRD.POST;
          NUM=NUM+1;
          GO TO READER;
          END;
                                          /*                                  */
                                          /* BLOCK ENTERED IF ITRIV=1 AT      */
                                          /* WRITER REQUEST OR NEW TERM THIS  */
                                          /* BINARY SEARCH IS WORD AFTER THE  */
                                          /* FIRST TERM READ                  */
ON01:     OLDWRD.OLDWORD=WRD.WORD;
          LOW=0;
          UPP=ILIMIT;
          DIV=UPP/2;
COMPAR:   IF DIV>SAVER THEN DO;
            UPP=DIV;
            GO TO COMPUT;;
            END;
          HOLD=STOP(DIV);
          IF OLDWORD<HOLD THEN DO;
            UPP=DIV;
            GO TO COMPUT;;
            END;
          IF OLDWORD>HOLD THEN DO;
            LOW=DIV;
            GO TO COMPUT;;
            END;
          GO TO TR;
COMPUT:   DIV=(LOW+UPP)/2;
          IF DIV ¬= LOW THEN GO TO COMPA ;
                        ELSE GO TO ON02;
TR:       ITRIV=1;
          GO TO READER;
ON02:     ITRIV=0;
          TOTAL=TOTAL+1;
          OLDWRD.POST(1)=WRD.POST;
          LPOST=WRD.POST;
          GO TO UNIQUE;
ENDPGM:
IF L0=1 THEN DO;
 L=L+1;
 L3=L3+K;
END;
NPOST=K;
OLDWRD.FREQ=NUM;
WRITE FILE(UNQWRD) FROM(OLDWRD);
    IF PTSW THEN CALL PRNT;
    TDUP=TDUP+DUP;  DUP=-1;
    IF PTSW THEN CALL PRNT;
        J=J+1;
        TOTAL=TOTAL+1;
        PUT EDIT('NUMBER OF UNIQUE WORDS:',TOTAL)(PAGE,A(23),F(8));
        PUT EDIT('NUMBER OF TRIVIAL WORDS:',NTRI)(SKIP(2),A(24),F(8));
        PUT EDIT('TOTAL POSTINGS:',J)(SKIP(2),A(15),F(10));
         PUT EDIT('DUPLICATE POSTINGS REMOVED:',TDUP)
              (SKIP,A,F(10));
```

```
      DJ=J;
      DL=TOTAL;
      AVERAGE=DJ/DL;
      PUT EDIT('AVERAGE NUMBER OF POSTINGS PER WORD:',AVERAGE)
        (SKIP(2),A(36),F(10,2));
       DD=TDUP;
       AVERAGE=DD/DL;
       PUT EDIT('AVERAGE NUMBER OF DUPLICATE POSTINGS:',
            AVERAGE)(SKIP,A,F(10.2));
J=J-L3;
PUT SKIP(5) EDIT('TOTAL LOW-FREQUENCY POSTINGS ',J)(A,F(8));
DJ=J;
DL=TOTAL-(NTRI+L2);
AVERAGE=DJ/DL;
PUT SKIP(2) EDIT('MEAN NUMBER OF POSTINGS PER LOW-FREQUENCY, NON-TRIVIA
L WORD ',AVERAGE)(A,F(10,5));
PUT SKIP(3);
PUT EDIT(L2,' UNIQUE HIGH-FREQUENCY WORDS',
       L, ' TOTAL RECORDS FROM H-F WORDS',
       L3,' TOTAL POSTINGS FROM H-F WORDS')((3)(SKIP(2),F(8),A));
      PUT EDIT('HIGHEST NUMBER OF POSTINGS PER WORD:',M)
       (SKIP(2),A(36),F(8));
   /* ****************************** ************************** */
                               /* THE PRNT SUBROUTINE  IS USED TO */
                               /*PRINT THE TERM FREQUENCIES IN A  */
                               /*THREE COLUMN PER PAGE FORMAT.    */
   PRNT:  PROC REORDER;
       DCL I1,I2 FIXED BIN(15);
       FLUSH: PROC;
            PUT PAGE EDIT(THL,THL,THL)(A(44),A(44),A(44));
            DO I1=1 TO 50;
                 PUT SKIP EDIT(PIM(I1))(A(132));
            END;
       END FLUSH;
       IF DUP<0 THEN DO;  /* FORCED FLUSH AT END OF RUN */
            DO I1=COL TO 3;
                 DO I2=LIN+1 TO 50;
                      SUBSTR(PIM(I2) (44*I1)-43,44)=' ';
                 END;
                 LIN=0;
            END;
            CALL FLUSH;
            RETURN;
       END;
       LIN=LIN+1;
       IF LIN>50 THEN DO;
            COL=COL+1;
            IF COL>3 THEN DO;
                 CALL FLUSH;
                 COL=1;
            END;
            LIN=1;
       END;
       PUT STRING(ITIM) EDIT(OLDWORD,NPOST,FREQ,FREQ+DUP)
            (A(20),(3)(F(7)));
       SUBSTR(PIM(LIN),(44*COL)-43,44)=ITIM;
   END PRNT;
   /* ********************************************************** */
      END SQUEEZ;
```

```
                                        /* LAST UPDATE: 750103 */
CLUSTER:
        PROC (RTP) REORDER OPTIONS(MAIN);
        DCL
             RTP CHAR(100) VAR,
 1 DISTNODE,
   2 TERM1 FIXED BIN(15),
   2 TERM2 FIXED BIN(15),
   2 TTDIST FLOAT DEC(6),
 WRITESW BIT(1) ALIGNED,
        SING BIT(1) ALIGNED STATIC INIT('1'B),
 OUT FILE RECORD,
        LNEC1 FIXED BIN(15) STATIC,
             LN(200) FIXED BIN(15) STATIC,
             ELTS(IIMAX,2) FIXED BIN(15) CTL,
             NXT(0:TOP) FIXED BIN(15) CTL,
             FORM(200,100) CHAR(1),
             FORMQ(1) CHAR(100) DEF FORM,
             (IIMAX,FIRST,LAST,CUR,EC1,EC2) FIXED BIN(15) STATIC,
             ASSOC(200)         DEC FLOAT(9), /* ASSOC WITH ABSORBER    */
             GROUP(200)         FIXED BIN(15),/* NO. OF ABSORBER        */
             SIZE(200)          FIXED BIN(15),/* SIZE OF GROUP          */
             DOCTRM(100,0:400) FIXED BIN(15),/* DOC-TRM COIN           */
             DOCDOC(0:5000)     DEC FLOAT(6), /* DOC-DOC ASSOC ARRAY    */
             CURR(100)          FIXED BIN(15),/*NO.OF CURR USER OF ROW */
            (I,DOCMAX,DOCNO,TCNT,TRMMAX,TRMNO,MULT1,MULT2,J,LOC,
             DOCCNT,
             T,INT,UN,HI,HJ,TOP,I2,BASX,BASI,BASJ,BASK,LOC2,LOC3,
             HHJ,HHI)           FIXED BIN(31) STATIC INIT(0),
            (XL,XU,DSTMIN,DIST,DIJ,DJK,DIK,DIX,ALPHJ,ALPHK,
             BETA,GAMMA)        DEC FLOAT(6) STATIC INIT(0),
             ROWMIN(100)        FIXED BIN(31),/* LOC:LOWEST DST IN ROW */
             ROWBASE(100)       FIXED BIN(31);/* ITEMS BEFORE ROW IN DD*/
                                        /*                              */
        IF INDEX(RTP,'WRITE')>0 THEN WRITESW='1'B;
        ELSE WRITESW='0'B;
        IF INDEX(RTP,'NOSING')>0 THEN SING='0'B;
        ELSE SING='1'B;
                                        /* NEAREST NEIGHBOR             */
        GAMMA=0;
                                        /* OTHER SETUP                  */
```

```
        DSTMIN=0;
        DO I=1 TO 100;
            CURR(I)=I;
        END;
        DOCTRM=0;
        DOCDOC=0;
        ROWMIN=0;
        DOCDOC(0)=2;
        SIZE=0;
        GROUP,ASSOC=0;
        INERR=0;
 ON UNDERFLOW BEGIN;
   PUT PAGE DATA(EC1,EC2,CUR,LAST,DIST,ID,LNEC1);
   PUT SKIP(2) DATA(TOP,DOCMAX);
   PUT SKIP(3) EDIT(LN(0),NXT(0))(F(5),F(5));
   GOTO DUMPPH;
 END;
        ON ERROR BEGIN;
        INERR=INERR+1; IF INERR>1 THEN GOTO EOP;
            CALL DUMP;
            GOTO DUMPPH;
        END;
        ON ENDFILE(SYSIN) GOTO STAGE2;

                                  /*                                  */
                                  /* READ DOC-TERM ARRAY              */
    RD:
        GET EDIT(DOCNO,DOCCNT,TCNT)(F(3),F(3),F(3));
  PUT SKIP EDIT(DOCNO,DOCCNT,TCNT)(F(3),X(2));
        IF DOCNO=0 THEN SIGNAL ENDFILE(SYSIN);
            SIZE(DOCNO)=DOCCNT;
        DO I=1 TO TCNT;
            GET EDIT(TRMNO)(F(3));
  PUT EDIT(TRMNO)(X(2),F(3));
            IF TRMNO=0 THEN DOCTRM(DOCNO,0)=DOCTRM(DOCNO,0)+1;
            ELSE DOCTRM(DOCNO,TRMNO)=1;
            IF TRMNO>TRMMAX THEN TRMMAX=TRMNO;
        END;
        IF DOCNO>DOCMAX THEN DOCMAX=DOCNO;
        GET SKIP;
        GOTO RD;
                                  /*                                  */
                      /* DOCTRM COMPLETE HERE; NOW DO DOCDOC          */
                                  /*                                  */
    STAGE2:

                                  /*                                  */
                      /* REMOVE REDUNDANT ROWS BY MERGING,            */
                      /* AND COMPRESS DOCTERM TO FILL HOLES           */
        TOP=DOCMAX;
        MULT1=2*DOCMAX;
        DSTMIN=2;                 /* LEGAL MAX IS 1                   */
        DO I=1 TO DOCMAX;
            MULT2=I-1;
            BASI=MULT2*(MULT1-I)/2;
            ROWBASE(I)=BASI;
                         /* THE DOC-DOC MATRIX IS STORED IN */
```

```
                                        /* REDUCED FORM. ONLY THOSE ITEMS  */
                                        /* DD(I,J) WHERE J>I ARE KEPT      */
                                        /* ITEM DD(I,J) IS IN DOCDOC(LOC)  */
                                        /* WHERE                           */
                                        /*        (I-1)(2N-I)              */
                                        /*  LOC= ----------- + J - I       */
                                        /*             2                   */
                                        /*                                 */
                DO J=I+1 TO DOCMAX;               /* DO A ROW OF DD        */
                     UN,INT=0;
           IF SING THEN UN=DOCTRM(I,0)+DOCTRM(J,0);
                     DO T=1 TO TRMMAX;
                          IF DOCTRM(I,T)>0 THEN
                               IF DOCTRM(J,T)>0 THEN
                                    INT=INT+DOCTRM(I,T)+DOCTRM(J,T);
                                         /* INT IS INTERSECTION SET SUM*/
                          UN=UN+DOCTRM(I,T)+DOCTRM(J,T);
                                         /* UN IS UNION SET SUM        */
                     END;
                     LOC=J-I+BASI;
                     INT=INT/2;
                     XI=INT;
                     UN=UN-INT;                /* DON'T COUNT TWICE     */
                     XU=UN;
                IF UN=0 THEN XU=1;
                     DIST=1-(XI/XU);
                     DOCDOC(LOC)=DIST;
                     IF DIST<DOCDOC(ROWMIN(I)) THEN ROWMIN(I)=LOC;
                     IF DIST<DSTMIN THEN DO;
                          DSTMIN=DIST;
                          HI=I;     HJ=J;      /* SAVE CLOSEST PAIR      */
                     END;
                END;
           END;
                                               /*                        */

PUT PAGE;
IF WRITESW THEN DO;
DO I=1 TO TRMMAX;
  DO J=I+1 TO TRMMAX;
    DISTSUM=0;  NUMDIST=0;
    DO I1=1 TO DOCMAX;
      IF DOCTRM(I1,I)=1 THEN DO;
        DO J1=1 TO DOCMAX;
          IF DOCTRM(J1,J)=1 THEN DO;
            NUMDIST=NUMDIST+1;
            IF J1>I1 THEN DISTSUM=DISTSUM+DOCDOC(ROWBASE(I1)+J1-I1);
       ELSE IF I1>J1 THEN DISTSUM=DISTSUM+DOCDOC(ROWBASE(J1)+I1-J1);
          END;
        END;
      END;
    END;
    TERM1=I; TERM2=J;  TTDIST=DISTSUM/NUMDIST;
    WRITE FILE(OUT) FROM(DISTNODE);
  END;
END;
```

```
  END;
                                   /* DOCDOC COMPLETE; NOW MAKE CLUSTERS    */
                                                  /*                        */
       IIMAX=2*TOP;
       ALLOCATE ELTS;
       ELTS=0;
    STAGE3:
       TOP=TOP+1;                       /* SET NEW GROUP NUMBER             */
       GROUP(CURR(HI)),GROUP(CURR(HJ))=TOP;
       ASSOC(CURR(HI)),ASSOC(CURR(HJ))=DSTMIN;
       PUT SKIP EDIT('FORMED ',TOP,' FROM ',CURR(HI),' , ',CURR(HJ),
            ' AT DISTANCE ',DSTMIN)
                  (A,F(3),A,F(3),A,F(3),A,F(7,5));
       IF SIZE(CURR(HI))>=SIZE(CURR(HJ)) THEN DO;
            ELTS(TOP,1)=CURR(HI);
            ELTS(TOP,2)=CURR(HJ);
       END;
       ELSE DO;
            ELTS(TOP,1)=CURR(HJ);
            ELTS(TOP,2)=CURR(HI);
       END;
       SIZE(TOP)=SIZE(CURR(HI))+SIZE(CURR(HJ));
       ALPHJ=SIZE(CURR(HI))/SIZE(TOP);
       ALPHK=SIZE(CURR(HJ))/SIZE(TOP);
       CURR(HJ)=0;                      /* THIS DELETES ROW HJ              */
       CURR(HI)=TOP;                    /* RE-USE ROW FOR NEW GROUP         */
       BASX=ROWBASE(HI);
       BASK=ROWBASE(HJ);
       DOCDOC(BASX+HJ-HI)=2;
                                                  /*                        */
                                  /* NOW FILL NEW DISTS IN ROW X            */
       DJK=DSTMIN;
       DSTMIN=2;
       ROWMIN(HI)=0;
       DO I=1 TO DOCMAX;                /* SET NEW DISTANCES TO ROW X       */
            IF CURR(I)=0 THEN GOTO SKIP;  /* A DELETED ROW                  */
            IF HI=I THEN GOTO SKIP;       /* SKIP ROW X ITSELF              */
            BASI=ROWBASE(I);
            DIST=2;
            IF I<HI THEN DO;
                 LOC2=BASI+HI-I;
                 DIJ=DOCDOC(LOC2);
                 IF LOC2=ROWMIN(I) THEN DIST=-1;
            END;
            ELSE DIJ=DOCDOC(BASX+I-HI);
            IF I<HJ THEN DO;
                 LOC2=BASI+HJ-I;
                 DIK=DOCDOC(LOC2);
                 DOCDOC(LOC2)=2;
                 IF LOC2=ROWMIN(I) THEN DIST=-1;
            END;
            ELSE DIK=DOCDOC(BASK+I-HJ);
            DIX=ALPHJ*DIJ+ALPHK*DIK+BETA*DJK+GAMMA*ABS(DIJ-DIK);
            IF I<HI THEN DO;
                 LOC=BASI+HI-I;
                 DOCDOC(LOC)=DIX;
```

```
                IF DIX<DOCDOC(ROWMIN(I)) THEN ROWMIN(I)=LOC;
          END;
          ELSE DO;
                LOC=BASX+I-HI;
                DOCDOC(LOC)=DIX;
                IF DIX<DOCDOC(ROWMIN(HI)) THEN ROWMIN(HI)=LOC;
          END;
          IF DIST<0 THEN DO;
                LOC3=BASI+DOCMAX-I;
                DIST=2;
                DO LOC2=BASI+1 TO LOC3;
                     IF DOCDOC(LOC2)<DIST THEN DO;
                          DIST=DOCDOC(LOC2);
                          ROWMIN(I)=LOC2;
                     END;
                END;
          END;
          IF DOCDOC(ROWMIN(I))<DSTMIN THEN DO;
                HHI=I;
                HHJ=ROWMIN(I)-BASI+I;
                DSTMIN=DOCDOC(ROWMIN(I));
          END;
SKIP:
     END;
     IF DOCDOC(ROWMIN(HI))<DSTMIN THEN DO;
          HHI=HI;
          HHJ=ROWMIN(HI)-BASX+HI;
          DSTMIN=DOCDOC(ROWMIN(HI));
     END;
     HI=HHI;
     HJ=HHJ;
     IF DSTMIN<2 THEN GOTO STAGE3;
                                   /*    */
     ALLOCATE NXT;
     FORM=' ';
     LN=0;
     NXT(TOP)=0;
     CUR,FIRST=TOP;
     LAST=0;
     DO WHILE(CUR¬=0);
          IF CUR>DOCMAX THEN DO;
               EC1,NXT(LAST)=ELTS(CUR,1);
               EC2,NXT(EC1)=ELTS(CUR,2);
               NXT(EC2)=NXT(CUR);
               DIST=(ASSOC(EC1)*10 )+.5;
               ID=DIST;
               FORMO(EC1),FORMO(EC2)=FORMO(CUR);
               IF LN(CUR)¬=0 THEN DO;
                    LNEC1,LN(EC1)=LN(CUR);
                    IF FORM(NXT(EC2),LNEC1)¬='*'
                         THEN FORM(EC2,LNEC1)=' ';
               END;
               ELSE LN(EC1)=ID;
               IF ID>0 THEN
               FORM(EC1,ID),FORM(EC2,ID)='*';
               LN(EC2)=ID;
```

```
                    CUR=EC1;
                END;
                ELSE DO;
                    LAST=CUR;
                    CUR=NXT(CUR);
                END;
            END;
                                        /*   */
            DO I=1 TO DOCMAX;
                DO J=1 TO LN(I);
                    FORM(I,J)='*';
                END;
            END;
            PUT PAGE;
            PUT EDIT('                ')(A(11));
            DO Q=.1 TO 1 BY .1;
                PUT EDIT(Q)(X(7),F(3,1));
            END;
            I=NXT(0);
            DO WHILE(I¬=0);
                PUT SKIP EDIT(I,FORMO(I))(F(4),X(6),A(100));
                I=NXT(I);
            END;
                                        /*   */
                                               /*                                   */
                                    /* WE SHOULD NOW BE ALL DONE                    */
                                               /*                                   */
      DUMPPH:
   PUT PAGE;
   PUT SKIP DATA(DOCMAX,TRMMAX);
   PUT SKIP(3) DATA(DSTMIN,HI,HJ,HHI,HHJ);
   J=((DOCMAX*DOCMAX)-DOCMAX)/2;
   PUT SKIP(3) DATA (J);
    IF DOCMAX=0 THEN DO;
           PUT SKIP;
   DO I=1 TO J;
     PUT EDIT(I,':',COCDOC(I))(X(3),F(4), (1),F(7,5));
   END;
   DUMP: PROC REORDER;
   PUT PAGE;
   DO I=1 TO DOCMAX;
    PUT SKIP EDIT(I,ROWBASE(I))(F(3),F(5));
     PUT EDIT(CURR(I))(X(3),F(4));
    PUT EDIT(SIZE(CURR(I)))(X(2),F(3));
    PUT EDIT(ROWMIN(I))(X(2),F(3));
           PUT SKIP;
    DO J=I+1 TO DOCMAX;
       PUT EDIT(J,':',DOCDOC(ROWBASE(I)+J-I))(X(3),F(4),A,F(7,5));
    END;
   END;
   END DUMP;
    END;
   PUT SKIP(3);
   DO I=1 TO TOP;
    PUT SKIP, EDIT(I,GROUP(I),ASSOC(I))(F(3),F(5),X(3),F(7,5));
       PUT EDIT(ELTS(I,1),ELTS(I,2),LN(I))((3)(X(3),F(5)));

     PUT EDIT(NXT(I))(X(3),F(5));
   END;
      EOP:
      CLUSTER;
```

APPENDIX D

WORD	1ST BIGGEST CASEC	%	2ND BIGGEST CASEC	%
AB	20	66	73	23
ABABA	53	100	0	0
ABATEMENT	60	100	0	0
ABDOMINAL	12	75	13	24
ABELIAN	70	100	0	0
ABERRATIONS	14	100	0	0
ABILITY	44	50	54	9
ABKHAZ	53	100	0	0
ABLATION	71	54	53	18
ABLE	16	100	0	0
ABNORMALITIES	14	100	0	0
ABOVE	77	33	70	16
ABRADABILITY	56	100	0	0
ABRASION	37	24	55	19
ABRASIVE	57	53	63	29
ABROAD	1	100	0	0
ABS	26	51	36	20
ABSCISIC	5	73	11	20
ABSENCE	35	37	36	19
ABSORBABILITY	39	100	0	0
ABSORBANT	49	100	0	0
ABSORBENT	49	34	50	29
ABSORBER	49	31	71	27
ABSORPTION	73	100	0	0
ABSORPTIOMETRIC	79	100	0	0
ABSORPTIONAL	47	100	0	0
ABSORPTIVITY	6	100	0	0
ABSTRACTION	67	35	70	19
ABUNDANCE	73	36	53	30
ABUSE	20	67	[illegible]	[illegible]
AC	5	100	0	0
ACADEMY	20	78	0	0
ACANTHASTER	12	100	0	0
ACARICIDE	5	63	27	24
ACCELERATED	71	43	75	33
ACCELERATION	71	36	76	13
ACCELERATOR	38	64	71	29
ACCEPTANCE	17	100	0	0
ACCEPTORS	22	100	0	0
ACCIDENTAL	8	75	71	23
ACCLIMATION	12	75	13	24
ACCLIMATIZED	1	100	0	0
ACCOMMODATION	65	100	0	0
ACCOMPANYING	54	41	79	17
ACCORDING	30	35	76	6
ACCOUNTS	45	100	0	0
ACCUMULATED	14	56	19	43
ACCUMULATOR	52	100	0	0
ACCURATE	79	55	0	0
ACEBUTOLOL	1	100	0	0
ACENAPHTHYLENE	25	100	0	0
ACER	3	100	0	0
ACERINOC	30	100	0	0
ACETABULARIA	10	100	0	0
ACETALDEHYDE	58	54	74	8
ACETAMIDE	23	35	39	31
ACETAMIDINES	1	100	0	0

D2

WORD	1ST BIGGEST CASEC	%	2ND BIGGEST CASEC	%
ACETAMIDOBUTADIENE	23	100	0	0
ACETAMIDOMETHYLCYCLO	32	100	0	0
ACETANILIDE	74	63	0	0
ACETATES	4	51	78	48
ACETAZOLAMIDE	1	100	0	0
ACETOACETAMIDE	27	100	0	0
ACETOACETAMIDOBENZIM	40	100	0	0
ACETOACETYL	13	100	0	0
ACETOHYDROXAMATE	3	100	0	0
ACETOIN	10	100	0	0
ACETONATE	76	100	0	0
ACETONEDICARBOXYLATE	27	100	0	0
ACETONITRILE	78	31	35	5
ACETONYLBENZYL	27	100	0	0
ACETOVANILLONE	22	100	0	0
ACETOXYALKOXY	29	100	0	0
ACETOXYBENZALDEHYDE	25	100	0	0
ACETOXYETHENE	36	100	0	0
ACETOXYHYDROXY	22	100	0	0
ACETOXYLATION	30	74	28	10
ACETYCHOLINE	12	100	0	0
ACETYLACETIC	23	100	0	0
ACETYLACETONATES	69	100	0	0
ACETYLACETONATOCARBO	35	100	0	0
ACETYLAMINOMETHYLCYC	32	100	0	0
ACETYLATED	34	53	35	[illegible]
ACETYLATORS	1	100	0	0
ACETYLBENZENE	72	100	0	0
ACETYLCELLULOSE	43	100	0	0
ACETYLCHOLINESTERASE	4	32	7	[illegible]
ACETYLCY[illegible]NTHENE	29	100	0	0
ACETYLDOPAMINE	12	100	0	[illegible]
ACETYLENEDICARBOXYLA	25	69	0	[illegible]
ACETYLENIC	32	27	29	[illegible]
ACETYL[illegible]	29	100	0	[illegible]
ACETYL[illegible]	19	100	0	[illegible]
ACETYL[illegible]	33	100	0	[illegible]
ACETYL[illegible]	35	100	0	[illegible]
ACETYL[illegible]DASE	7	100	0	[illegible]
ACETYL[illegible]	25	100	0	[illegible]
ACETYL[illegible]	76	100	0	[illegible]
ACETYLTRANSFERASE	3	54	6	[illegible]
ACETYLTRI[illegible]	22	100	[illegible]	[illegible]
[illegible]	12	[illegible]	79	37
[illegible]	3	100	[illegible]	[illegible]
[illegible]	[illegible]	53	14	[illegible]
[illegible]	50	100	0	[illegible]
[illegible]	22	17	66	[illegible]
[illegible]	51	100	0	[illegible]
[illegible]	12	100	[illegible]	[illegible]
[illegible]	45	[illegible]	25	[illegible]
[illegible]	19	100	0	[illegible]
ACOUSTIC	65	40	74	[illegible]
[illegible]	50	100	0	[illegible]
[illegible]	70	100	0	[illegible]
[illegible]	57	77	73	[illegible]
[illegible]	75	100	0	[illegible]

WORD	1ST BIGGEST CASEC	%	2ND BIGGEST CASEC	%
ACQUISITIONS	16	100	0	0
ACRIDINE	27	37	75	22
ACRIDONE	27	86	73	13
ACROMEGALY	1	100	0	0
ACRONYCINIUM	31	100	0	0
ACRYLAMIDE	17	22	43	15
ACRYLAMIDOANTHRAQUIN	35	100	0	0
ACRYLATED	42	100	0	0
ACRYLONITRILE	35	27	39	24
ACRYLOYLOXYBENZYLIDE	35	100	0	0
ACS	20	100	0	0
ACTH	2	87	13	12
ACTIN	5	49	7	29
ACTINIDE	20	63	71	36
ACTININ	6	100	0	0
ACTINOMYCES	10	71	0	0
ACTINOMYCIN	1	54	15	30
ACTINOXANTHIN	6	100	0	0
ACTION	3	17	3	12
ACTIVATE	2	100	0	0
ACTIVATING	67	100	0	0
ACTIVATOR	38	41	74	36
ACTIVITIES	18	14	46	14
ACTOMYOSIN	7	58	17	41
ACULEATUS	2	100	0	0
ACUTE	2	25	14	14
ACYLACRONYCINIUM	31	100	0	0
ACYLAMIDASE	4	100	0	0
ACYLAMINO	62	72	0	0
ACYLAMINODEOXY	33	100	0	0
ACYLAMINOMETHYL	24	100	0	0
ACYLAMINOPHENYL	25	100	0	0
ACYLARYLSULFONAMIDE	3	100	0	0
ACYLATED	35	69	0	0
ACYLATING	34	100	0	0
ACYLAZIRIDINE	22	100	0	0
ACYLBENZIMIDAZOLE	25	100	0	0
ACYLFERROCENE	29	100	0	0
ACYLGLUCOSAMINE	13	100	0	0
ACYLHYDRAZINOANILINE	28	100	0	0
ACYLNITRAMINES	25	100	0	0
ACYLOXY	27	63	29	31
ACYLOXYMETHYL[illegible]	29	100	0	0
ACYLOXYSILANE	29	100	0	0
ACYLPHENYLDIAZOMETHA[illegible]	27	100	0	0
ACYLTHIOCARBOXYMETHYL[illegible]	27	100	0	0
[illegible]	1	100	0	0
[illegible]	1	100	0	0
ADAPT	20	100	0	0
ADAPTED	65	51	0	0
ADAPTMENTS	1	100	0	0
ADD	22	100	0	0
ADDICTING	20	100	0	0
ADDING	52	97	0	0
ADDITION	31	15	25	16
ADDITIONALLY	72	100	0	0
ADDITIV	55	100	0	0

BIBLIOGRAPHY

BIBLIOGRAPHY

1. E. Hilgard and B. Bowers; Theories of Learning, Prentice Hall, 1975.

2. E. Garfield, M. V. Malin and H. Small; A System for Automatic Classification of Scientific Literature, J. Indian Institute of Science, V57, N2, pp. 61-74, 1975.

3. G. Salton; Automatic Information Organization and Retrieval, McGraw Hill, New York, 1968.

4. K. Sparck-Jones and M. Kay; Linguistics and Information Science, Academic Press, New York, 1973.

5. A. Montgomery, Linguistics and Information Science, J. ASIS, May/June, 1972, pp. 195-219.

6. F. J. Damerau; Automated Language Processing, in: Annual Review of Information Science, V11, M.E. Williams, Editor, A.S.I.S., 1976, 457 pp.

7. N. Sager; Evaluation of Automated Natural Language Processing in the Further Development of Information Retrieval, String Program Report No. 10, NYU Linguistic String Project, July, 1976.

8. S. Kuno and A. G. Oettinger; Multiple Path Syntactic Analyzer, Information Processing, 1962, Amsterdam, North-Holland Publishing Co.

9. A. M. Zwicky, J. Friedman, B. C. Hall and D. E. Walker; The MITRE Syntactic Analysis Procedure for Transformational Grammars, Proceedings Fall Joint Computer Conference, Washington, DC, Spartan Books, 1965, pp. 317-326.

10. Y. Wilks; Computible Semantic Derivations, System Development Corp., California, Jan. 1968, 160 pp. (SP-3017).

11. J. Laffal; Total or Selected Content Analysis, International Conf. on Computational Linguistics, Preprint No. 24, 1969.

12. E. Von Glaserfeld; Semantics and the Syntactic Classification of Words, Int. Conf. on Computational Linguistics, Preprint No. 22, 1969.

13. R. F. Simmons; Inferential Question Answering in a Contextual Data Base, pp. 51-56, Proceedings Annual Conference of the ACM, Houston, TX, Oct. 1976.

14. R. C. Schank; Conceptual Information Processing, Amsterdam, North-Holland, 1975.

15. T. Winograd; Understanding Natural Language, Academic Press, New York, 1972.

16. G. Salton; The SMART Retrieval System, Prentice Hall, 1971.

17. Karen Sparck-Jones; Automatic Keyword Classification and Information Retrieval, Butterworth, London, 1971.

18. C. T. Yu and G. Salton; Precision Weighting - An Effective Automatic Indexing Method, J. ACM, V23, pp. 76-88, 1976.

19. S. E. Robertson, K. Sparck-Jones; Relevance Weighting of Search Terms, J. ASIS, May/June, 1976, pp. 129-146.

20. Anonymous; Subject Coverage and Arrangement of Abstracts by Sections in Chemical Abstracts, Chemical Abstracts Service, Ohio State University, OH, 1975.

21. Anonymous; Card-A-Lert The Selective Engineering Information Service, Engineering Index, Inc., New York, 1972.

22. R. H. A. Sneath, R. R. Sokal; Principles of Numerical Taxonomy, Freeman, San Francisco, 1963.

23. M. Lance, W. R. Williams; A General Theory of Classification Sorting Strategies, Computer J., V9, pp. 373-380, 1970.

24. J. N. Lance, W. T. Williams; A General Theory of Classificatory Sorting Strategies II, Clustering Systems, Computer J., V10, pp. 173-177, 1970.

25. J. Van Rijsbergen; Further Experiments with Hierarchic Clustering in Document Retrieval, Information Storage and Retrieval, V10, pp. 1-14, 1974.

26. P. Atherton; Bibliographic Data Bases - Their Effect on User Interface Design in Interactive Retrieval Systems In: D. E. Walker - Editor, Interactive Bibliographic Search, AFIPS Press, 1971.

27. S. E. Preece; The Use of Hierarchic Clustering in Forming a Similarity Measure, presented at the 12th Annual Allerton Conference on Switching and Circuit Theory, 1974.

28. C. Landauer, T. Morris, et al; Application of Pattern Analysis and Recognition to I&W, PAR Report No. 76-3, New York, Jan. 1976.

29. S. P. Harter, A Probabilistic Approach to Automatic Keyword Indexing Part I. On the Distribution of Specialty Words in a Technical Literature, J. ASIS, July/Aug. 1975, 00. 197-206.

30. G. Salton, A. Wong and C. T. Yu; Automatic Indexing Using Term Discrimination and Term Precision Measurements, Information Processing and Management, V12, pp. 43-51, 1976.

31. M. E. Williams, Criteria for Evaluation and Selection of Data Bases and Data Base Services, Special Libraries, V66, pp. 561-569, 1975.

32. E. M. Onderisin; The Least Common Bigram: A Dictionary Arrangement Technique for Computerized Natural-Language Text Searching, Proceedings of Annual Conference of ACM, 1971, pp. 82-96.

33. D. E. Meyer and R. W. Schvaneveldt; Meaning, Memory Structure and Mental Processes, Science, V192, pp. 27-33, 2 April 1976.