DOCUMENT RESUME

ED 206 286 IR 009 549

AUTHOR Hickey, Thomas B.
TITLE Research Report on Development of a Probabilistic Author Search and Matching Technique for Retrieval and Creation of Bibliographic Records.
INSTITUTION OCLC Online Computer Library Center, Inc., Dublin, Ohio.
SPONS AGENCY National Science Foundation, Washington, D.C.
REPORT NO OCLC/OPR/RR-81/2
PUB DATE 27 May 81
GRANT IST79-18263
NOTE 48p.

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Algorithms; *Authors; Bibliographies; Computational Linguistics; *Databases; *Information Retrieval; *Online Systems; Statistical Analysis

ABSTRACT

Using macro and micro-structure analysis of large files of personal author names, this study developed retrieval techniques and algorithms to automatically correct and/or flag typographical errors in names, identify names in a database that are similar to a name entered by a user during a search, and measure similarities between names. It was found that personal names have very different characteristics than English language words, and this project demonstrated that useful displays for human verification of author names can be built, although at some computational expense. Automatic correction of errors, requiring greater computation, was not demonstrated by the project; however, such correction seems feasible with extensions of the techniques developed for automatic detection. A bibliography of 39 titles is included. (Author/RAA)

Report Number:OCLC/OPR/RR-81/2
Date: 1981 May 27

Research Report

on

Development of a Probabilistic Author Search and Matching Technique for Retrieval and Creation of Bibliographic Records

Supported by: The National Science Foundation

Grant Number: IST79-18263

Principal Investigator: Thomas B. Hickey Ph.D, Research Scientist

OCLC
Office of Planning and Research
Research Department
6565 Frantz Road
Dublin, OH 43017

The Research Report Series is OCLC's formal dissemination vehicle through which OCLC research project results can be made public.

The Research Department's Research Reports are published and distributed through OCLC, until the report is available from ERIC (approximately six to nine months after publication). A maximum of three copies of any single Research Report is provided at no charge to interested persons and institutions. Requests for Research Reports should be directed to OCLC, User Services Division, Customer Services Section, 6565 Frantz Road, Dublin, OH 43017.

Research Report Series

Subject Heading Patterns in OCLC Monographic Records
by E. O'Neill and R. Aluri
Report Number: OCLC/RDD/RR-79/1; ERIC ED 183 167

An Overview of a Proposed Monitoring Facility for the Large-scale, Network-based OCLC On-line System
by W. Dominick, D. Penniman, and J. Rush
Report Number: OCLC/OPR/RR-80/1; ERIC ED 186-042

Analytical Review of Catalog Use Studies
by K. Markey
Report Number: OCLC/OPR/RR-80/2; ERIC ED 186 041

A Method for Correcting Typographical Errors in Subject Headings in OCLC Records
by E. O'Neill and R. Aluri
Report Number: OCLC/OPR/RR-80/3

Field, Subfield, and Indicator Statistics in OCLC Bibliographic Records
by T. B. Hickey
Report Number: OCLC/OPR/RR-81/1

Development of a Probabilistic Author Search and Matching Technique for Retrieval and Creation of Bibliographic Records
by T. B. Hickey
Report Number: OCLC/OPR/RR-81/2

ABSTRACT

This study undertook an in-depth analysis of large files of personal author names to permit the development of techniques and algorithms to automatically:

(1) correct and/or flag typographical errors in names,
(2) identify names in a data base that are similar to a name entered by a user during a search, and
(3) measure similarities among names.

The study found that personal names have very different characteristics than English language words. This project demonstrated that useful displays for human verification of author names can be built, although at some computational expense. Automatic correction of errors, which would require even greater computation, was not demonstrated by this project. However, automatic correction seems feasible with extensions of the techniques in this project for automatic detection.

ACKNOWLEDGEMENTS

The author acknowledges the efforts of staff in the Data Base Administrative Section, Computer Systems Engineering Department, OCLC, in extracting the sample of records used in this report. In addition, the author thanks Peggy Zimbeck and Larry Morwick for their editorial assistance.

NOTE ABOUT THE AUTHOR

Dr. Thomas B. Hickey has been with OCLC for over four years. He is currently a Research Scientist in the Research Department. Dr. Hickey holds a B.S. in Physics from the State University of New York (SUNY) at Stony Brook, an M.L.S. from SUNY at Geneseo, and a Ph.D. in Library Science from the University of Illinois. His current areas of interest at OCLC include automatic matching of author names for authority control and electronic document delivery.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Report Number:OCLC/OPR/RR-81/2
Date: 1981 May 27

I. INTRODUCTION

In 1979 March, the National Science Foundation (NSF) awarded OCLC a grant to study the "Probabilistic Matching and Control of Author Names in Automated Library Systems." The grant for $42,321 was to cover the period 1979 August 15 through 1981 January 31. The grant, #IST79-18263, was awarded as part of the Information Science Unit's "New Investigators in Information Science Special Research Initiation Awards."

The principal investigator of this project was Dr. Thomas B. Hickey, Research Scientist, OCLC. K.B. Rastogi, Research Scientist; Ronald Ringenberg, Richard Tobin, and Christopher Picone, Research Assistants, comprised the project team.

A. Research Objectives

The objectives of this study, as outlined in the proposal, were to develop techniques and algorithms that automatically:

(1) correct and/or flag typographical errors in names,
(2) identify from a data base, names similar to those entered by a user during a search,
(3) measure similarities among names.

In essence, the project goal was to match and control names automatically. This matching and control is currently performed manually, if at all, using authority files constructed by librarians.

B. Project Organization

The research concentrated primarily on the identification and matching of surnames. Some work was done on extending results to other parts of names and other personal information contained in bibliographic records, such as dates and titles (e.g., Dr., Mrs.); however, these efforts were minimal. The study of surnames proceeded in three phases.

In Phase 1 the micro- and macrostructures of names were analyzed. This analysis is presented in Section IV of this report. Microstructure research focused on breaking names into smaller units such as bigrams (two-letter pairs), trigrams (three-letter groupings), or syllables. Macrostructure studies investigated the overall characteristics of surnames in the OCLC data base, such as the type/token distribution.

In Phase 2, techniques were developed that can identify names "similar" to a given name being searched. The project team developed both a retrieval system based on character structure and a clustering system for name comparison based on distance measures.

The distance measure, frequency information, and other name information formed the basis for Phase 3. This phase, discussed in Section VI, ranked names by the probability that they are the same.

C. Accomplishments

The most important result of the project is the demonstration that, in a given language, personal names are very different from words in nearly every characteristic. The differences that most affect the development of retrieval algorithms are:

(1) the very large number of unique names,
(2) the evenness of bigram and trigram distributions, and
(3) the lack of uniformity in structure.

These differences seem to occur because of the extremely diverse linguistic background under which names have evolved. Although some affix and suffix structures can be observed, the diversity of origins frustrates the straightforward techniques that can be applied with some success with English words (see RESNIKOFF).

D. Conclusions

This project has demonstrated that useful displays for human verification of author names can be built, although at some computational expense. Automatic correction of errors, which would require even more computation, was not demonstrated by this project. However, automatic correction seems entirely feasible with only slight extensions of the techniques developed in this project for automatic detection.

E. Future Directions

A fully automatic authority system is beyond the reach of present computer systems. Large amounts of computer resources would be required to scale these experiments up to fully operational data bases of millions of records and thousands of names to be verified each day. The OCLC Online System, for example, would have to verify a name approximately every five seconds to keep up with new bibliographic records being entered. An automatic authority system can be envisioned, but the system, implemented with little additional overhead to the checking now performed on bibliographic records, would require hardware capabilities beyond those available today.

II. BACKGROUND AND LITERATURE REVIEW

Traditionally in libraries, vagaries in authors' names have been controlled manually with the use of authority files. Authority files contain information that relates variant forms of an author's name as well as publication dates and subject areas used to distinguish authors with identical names. Authority files list problem cases with which the creators of the files are familiar, and the prescribed resolution of these cases. The creation of authority files involves developing extensive networks of cross-references and extensive human effort to maintain a consistent catalog.

Computers are sometimes involved in this effort, although to date this involvement has been minimal. Primarily, computers are used to maintain manual authority files. Any name-matching done by the computer is of the simplest kind, i.e., looking for the exact matches of names, or parts of names, and displaying the result of such a search/match to the user.

With the rapid development of computer hardware there is increased interest in having programs take over as much of the human effort as possible. SHARPE describes an excellent example of the type of application possible in a library situation, in this case a computer-assisted authority system for Chemical Abstracts Services indexes. LEE takes the idea further by having a computer perform a cluster analysis on data, looking for anomalous records.

The problem faced by libraries is part of the larger problem of error correction by the approximate matching of strings. During the course of this research two excellent review articles have appeared (PETERSON and HALL), so that I will only review those articles with specific application to this research.

In general, the work on string-matching has limited application to authority work. Little research has dealt with names, and none with the millions of surnames that are encountered in large library data bases. There has, however, been a fair amount of research on the microstructure of words and names and its application to information retrieval. FOKKER studied surnames, using variable-length strings, which divide authors into uniform frequency distribution. This is the only large-scale investigation of names (up to 100,000) in the literature. Others, such as those by DEHEER and WILLETT, have employed the information inherent in substrings of terms for promising experiments in subject retrieval.

The microstructure of words has also been employed in error correction, most notably by ULLMAN and RISEMAN whose technique is more fully explained in Section IV. Microstructure was also employed in an early study by CARLSON, which is especially interesting because it was used with names, although from a very homogeneous population.

In addition to the simple microstructure reflected in n-grams, for information retrieval using subject terms, it has been found to be useful to break words into their roots and affixes. This study experimented with application of this technique to names using hyphenation algorithms similar to those by RICH and MOITRA and developing an algorithm similar to that reported by HAFER, which requires comparison of each word to words surrounding it alphabetically.

Once candidate names have been retrieved, a method for ranking them by similarity is needed. Most investigators, e.g., TAGLIACOZZO, note that the majority of errors can be classified as replacement, omission, addition, or transposition errors. WAGNER and LOWRANCE give rigorous measures of string similarity based on these transformations and the minimum number of simple edit operations needed to change one string to another. In this study, however, an algorithm developed for file comparisons by HECKEL was extended to give a measure which had the important property of recognizing the common name variation of an inverted multiple surname.

III. METHODOLOGY

A. Description of the Data Base

The source data base used in this study was OCLC's Online Union Catalog. The Online Union Catalog presently consists of over 7.2 million bibliographic records, with 25,000 new records being added weekly. Each bibliographic record represents a book, serial, or various other piece of library materials cataloged online by a member library or batch entered from MARC tapes. On the average, each bibliographic record includes 10.5 location symbols that show which OCLC member libraries hold that item. Holdings information was not used in the results reported.

From the Online Union Catalog, three major samples of records were drawn for the study. The first was a random 1% sample of the full data base (41,840 names) at the time the sample was drawn (1978 September 2). The second sample consisted of all records containing a publication date of 1976. This sample (drawn initially for another project) contained 343,593 bibliographic records, about 5% of the full data base at the time it was drawn (1980 February through March). The third sample contained all records that had a personal author surname beginning "SM." This sample contained 38,658 records and was drawn in the summer of 1980. Table 1 summarizes the sample characteristics.

Table 1. Characteristics of the Three Samples

Sample	Number of Records	Number of Names	Characteristics
1% (1% of Data Base)	41,212	41,840	random sample
1976 (5% of Data Base)	343,593	344,183	all records containing a publication date of 1976
SM (<1% of Data Base)	38,658	58,191	all records that had a personal author surname beginning "SM"

From these samples all personal names occurring in the 100 (main entry, personal name) and 700 (added entry, personal name) fields were extracted. The names in these fields were then subjected to a fairly extensive editing and normalization procedure that eliminated diacritics, converted lowercase to uppercase, converted ligatures to their two-character equivalents, and collapsed certain variant forms of letters, such as a script L and Turkish I to their more commonly used counterparts.

To determine name variations, a sample of changes made to the Online Union Catalog was collected. These changes consisted of error reports for names corrected by OCLC's Bibliographic Record Management Group over a period of two weeks in the fall of 1980. Because of the importance of such a sample for work on automated authority files, these names and corrections are presented in Appendix A.

B. Programming Techniques

Most of the name microstructure studies were programmed using FORTH. The FORTH programming language is nearly unique in that it offers highly interactive program development support while retaining reasonable execution efficiency. In addition, FORTH is very easy to interface to the underlying hardware for special needs. For this project, a facility for multidimensional bit arrays proved invaluable in the bigram and trigram studies. This bit vector approach to file inversion was used as suggested by KING and LEFKOVITZ. Unfortunately, the resultant FORTH source code is difficult to follow--impossible for one who has not programmed in the language.

The cluster-building and searching programs as well as the name-matching decision tables were written in PASCAL. PASCAL is becoming a very widely used programming language because it provides excellent data structures and highly readable code. For cluster searching a rather unique technique was used: linked lists based on those used by LISP were programmed along with the classic LISP operators for such tasks. (See HENDERSON for the LISP structures and algorithms used.) It appears likely that this linked list package in PASCAL could have wider applications for sophisticated text processing.

One task attempted in the project was to convert a negative binomial curve-fitting package written in CDC FORTRAN to run on OCLC's Sigma 9 computers. It was not possible, however, to eliminate conversion problems that were evidently the result of the different precision and range of the floating point representations on the two computers. In the future, the conversion may be completed after the author of the curve-fitting package, Dr. Edward O'Neill, Dean, Case Western Reserve University, School of Library Science, completes conversion of the CDC FORTRAN program to a DEC System 20.

IV. MACROSTRUCTURE AND MICROSTRUCTURE STUDIES

A. Macrostructure

Macrostructure statistics include all of those statistics gathered that were not obtained by breaking up the individual words or names. These statistics are primarily length statistics, frequency distributions, and extrapolations to larger files. Throughout the project, the emphasis was on surnames, since forenames are often unavailable.

1. Length Statistics

The 1% sample had an average of 1.02 personal names in each bibliographic record. Each personal name contained an average of 2.6 separate parts. The full name occupied an average of 19.1 characters.

The average length of author surnames was 7.0 characters with a standard deviation of 2.8 characters. Figure 1 presents a length distribution for personal surnames. Of these surnames, (..% were "multiple surnames"; that is, they had more than one part.

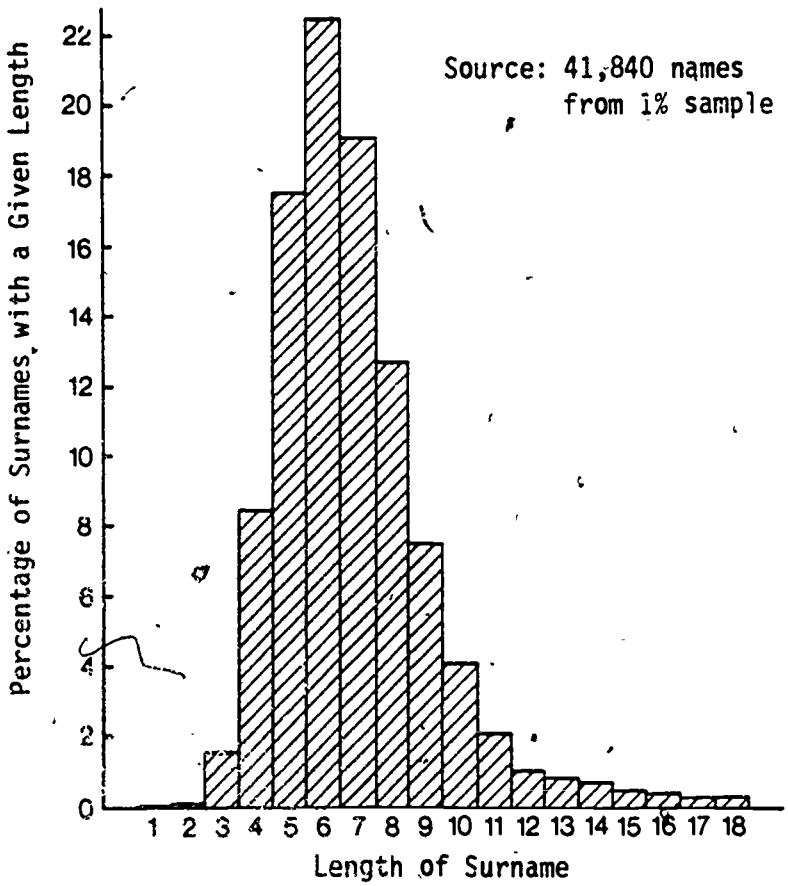

Figure 1. Surname Length Distribution

Forenames (including those consisting only of initials) averaged 5.6 characters with a standard deviation of 1.9 characters. Figure 2 shows the length distribution of forenames. The peak at a length of one (4.4%) is caused by names having only an initial.

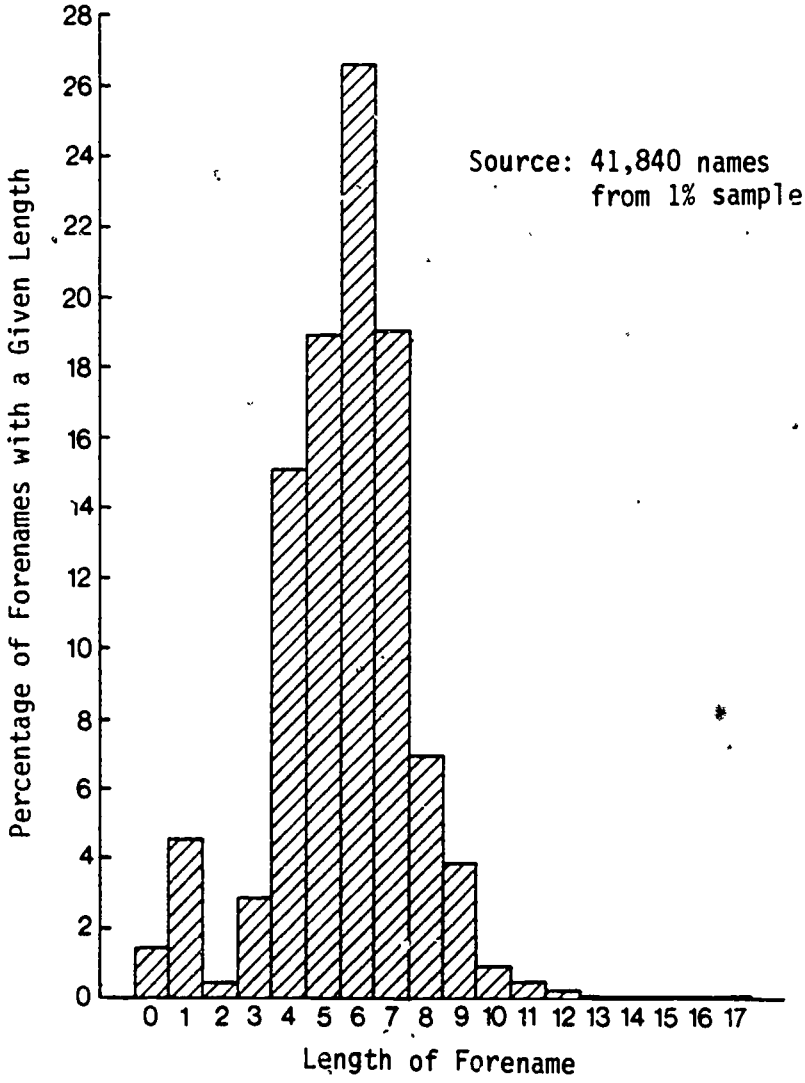

Figure 2. Forename Length Distribution

2. Frequency Distributions

Frequency of occurrence statistics were gathered for the 1% sample and the SM sample. Table 2 lists the 100 most frequently occurring surnames from the 1% sample along with their frequency. "SMITH" is by far the most popular name, followed by "JOHNSON" which contributes less than half as many. It should be pointed out, however, that "SMITH" accounts for only 229 names out of the 41,840 names in the sample (0.55%). The listing in Table 2 of the 100 most-common authors is obviously different than one would expect from random surnames; e.g., "BACH" ranks 17th, "SHAKESPEARE" ranks 22nd, and "MOZART" ranks 25th.

Table 2. The 100 Most Common Author Surnames

Rank	Name	Frequency	Rank	Name	Frequency
1	Smith	(229)	51	Nelson	(36)
2	Johnson	(109)	52	Muller	(35)
3	Jones	(103)	53	Edwards	(35)
4	Brown	(100)	54	Cohen	(35)
5	Miller	(98)	55	Simon	(34)
6	Williams	(96)	56	Rogers	(34)
7	Wilson	(84)	57	Phillips	(34)
8	Taylor	(82)	58	Mitchell	(34)
9	Anderson	(72)	59	Meyer	(34)
10	Davis	(71)	60	Walker	(33)
11	Wright	(69)	61	Richardson	(33)
12	Clark	(69)	62	Turner	(32)
13	Thomas	(62)	63	Morris	(32)
14	Hall	(62)	64	Haydn	(32)
15	Thompson	(62)	65	Graham	(32)
16	Scott	(61)	66	Cox	(32)
17	Bach	(60)	67	Morgan	(31)
18	Adams	(60)	68	Kelly	(31)
19	Lewis	(59)	69	Ford	(31)
20	Martin	(58)	70	Davies	(31)
21	White	(57)	71	Carter	(31)
22	Shakespeare	(57)	72	Stevens	(30)
23	Allen	(57)	73	May	(30)
24	Moore	(56)	74	Knight	(30)
25	Mozart	(55)	75	Howard	(30)
26	King	(54)	76	Wells	(29)
27	Baker	(52)	77	Watson	(29)
28	Robinson	(50)	78	Johnston	(29)
29	James	(48)	79	Stone	(28)
30	Young	(46)	80	Reynolds	(28)
31	Roberts	(46)	81	Porter	(28)
32	Green	(46)	82	Gordon	(28)
33	Russell	(45)	83	Gardner	(28)
34	Harris	(45)	84	Butler	(28)
35	Lee	(43)	85	Bell	(28)
36	Hill	(43)	86	Bailey	(28)
37	Ward	(42)	87	Shaw	(27)
38	Campbell	(41)	88	Price	(27)
39	Beethoven	(41)	89	Lawrence	(27)
40	Cook	(40)	90	Harrison	(27)
41	Wagner	(39)	91	Gray	(27)
42	Jackson	(39)	92	Fisher	(27)
43	Cooper	(39)	93	Dickens	(27)
44	Wood	(38)	94	Be.. tt	(27)
45	Parker	(38)	95	Andrews	(26)
46	Hamilton	(38)	96	Stevenson	(25)
47	Evans	(38)	97	Palmer	(25)
48	Weber	(37)	98	Myers	(25)
49	Stewart	(37)	99	Murphy	(25)
50	Murray	(37)	100	Mason	(25)

Figure 3 is a graph depicting the 100 most frequently occurring names listed in Table 2. The points used to plot the curve in Figure 3 were rather irregular, especially at the highest ranking points (SMITH, JOHNSON, etc). Looked at from another point of view, however, the curve smooths out and proves more amenable to analysis.

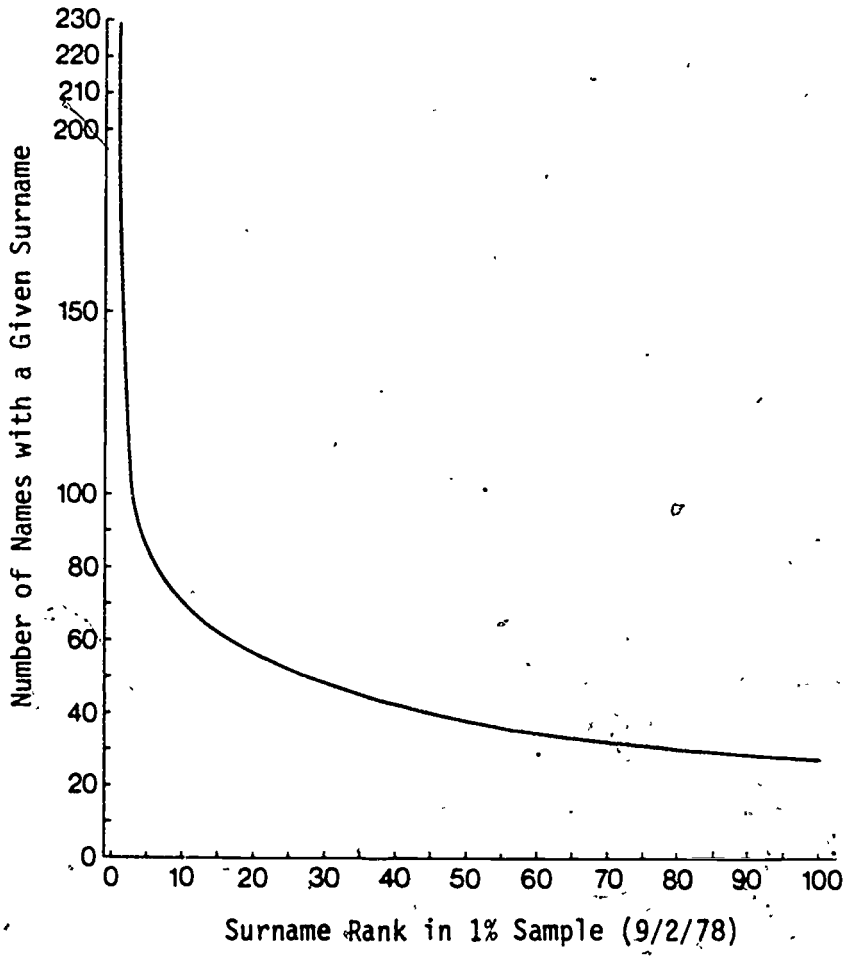

Figure 3. The 100 Most-Common Author Surnames

Figure 4 (p. 11) is a plot of the frequency of each type of surname (229 for SMITH, 31 for DAVIES) versus how many types have that frequency (only one group has 229 members, but five groups have 31 members and 16,879 names only occurred once). This plot shows the least squares fit to the log-log curve for the 1% sample, a random 1/16th of the 1% sample, and the 1976 sample. All curves are very regular and remarkably uniform, considering that the file sizes span two orders of magnitude, ranging from 2,600 to over 300,000.

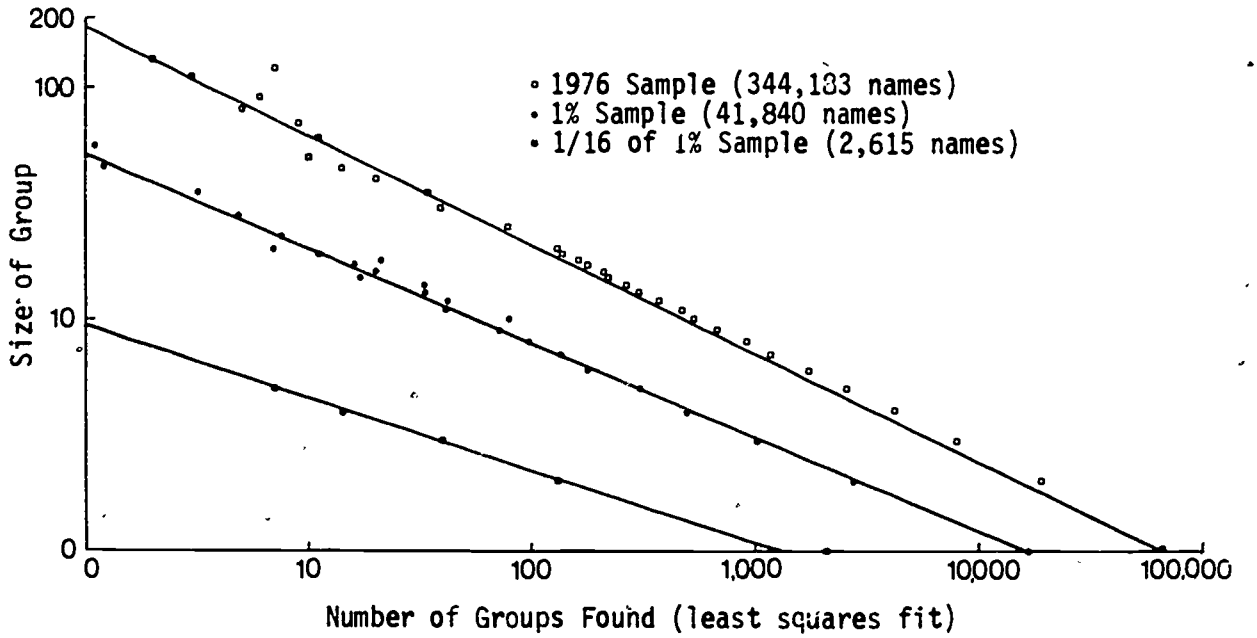

Figure 4. Surname Frequency Distribution for Various File Sizes

Figure 5 (p. 12) shows a plot of the number of unique surnames of a series of subsamples of the 1% sample (1/16, 1/8, 1/4, 1/2) along with the first 1/3 of the 1976 sample and the full 1976 sample. Only when the full (large) 1976 sample is plotted, is there any leveling off of the curve. This leveling (remembering that it is shown in a full logarithmic plot) indicates that the file will tend to reach a "saturation" point where the number of unique surnames in the file will occupy a slightly lower percentage of all surnames.

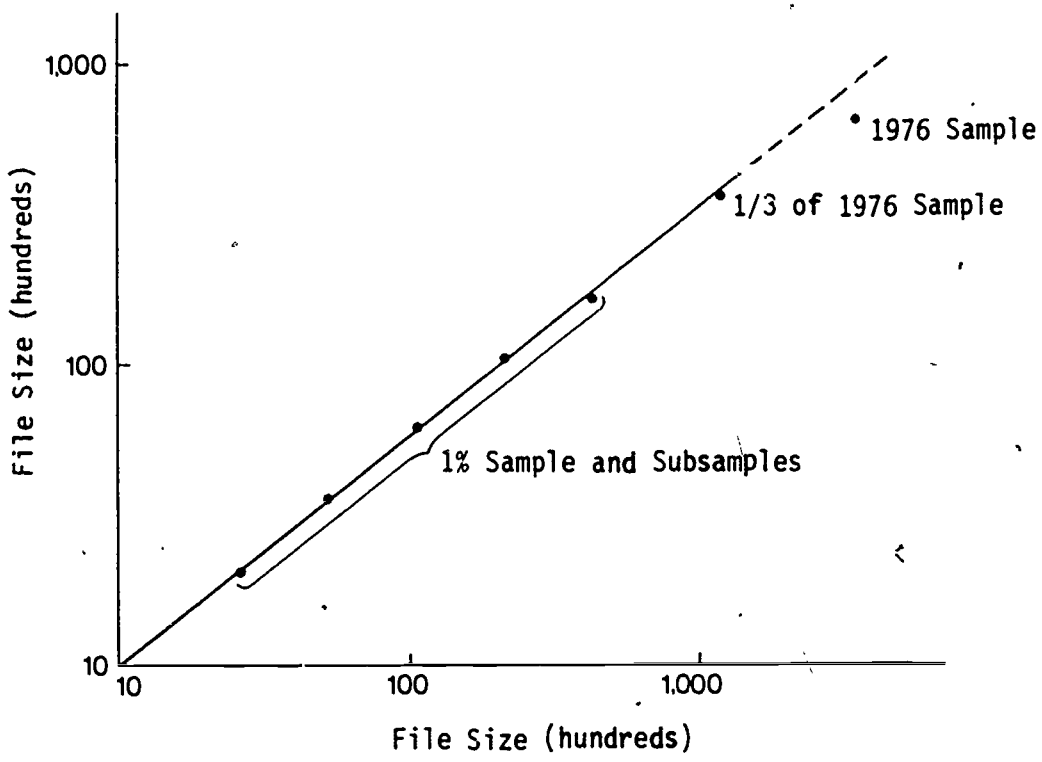

Figure 5. Surnames Recurring Once vs. File Size

3. Extrapolations to Larger Files

One of the more important statistics on file distribution is the number of unique surnames contained in the full file. Dr. Edward O'Neil has implemented a curve-fitting program based on the negative binomial distribution. This software is written in CDC FORTRAN. Unfortunately, therefore, the software could not be run successfully on the OCLC Sigma 9 hardware for the large sample sizes.

Figure 6 (p. 13) is a plot of the predicted number of unique surnames found versus file size. The figure shows the 1% sample, its subsamples, the first third of the SM sample, and the full 1976 sample, again plotted on a full-logarithmic scale. While extrapolation from these statistics must be carefully assessed, they imply that OCLC's present file of over 7 million bibliographic records contains somewhere between 700,000 and 1.4 million unique surnames.

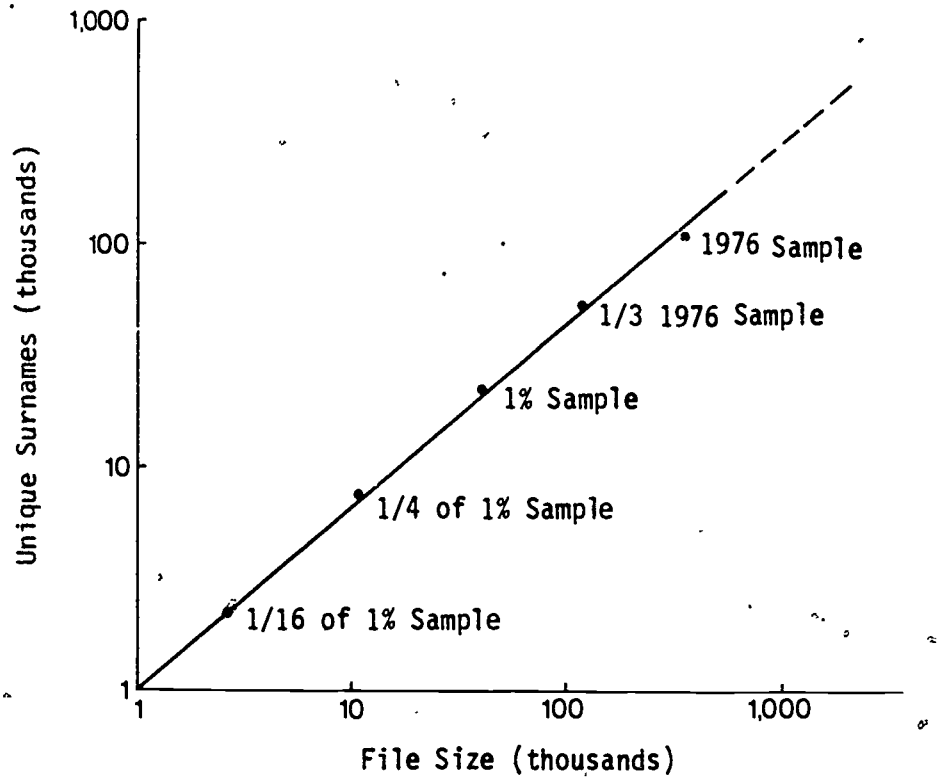

Figure 6. Predicting Number of Unique Surnames

The SM sample* offers another method of estimating the number of unique surnames in the full file (5.75 million records at the time the sample was drawn). The SM sample contains 779 different surnames beginning with "SM," compared with 25 different surnames beginning with "SM" found in the 1% sample. Assuming this ratio 779/25 holds for other sections of the file, this gives the following equations:

779/25 X 22,385 (the number unique in the 1% sample) = 700,000 names
versus
620,000 to 1,200,000 for 5.75 million records predicted by Figure 6.

The above assumption that the ratio is representative of the whole file, however, has not been tested.

This assumption yields a very large number of different surnames (several times the size of the number of words in the English language) and has significant consequences when attempting to "normalize" names; i.e., bring variant forms together. In particular, the large number implies much less regularity in the microstructure of names than is found in typical words. The following section on microstructure confirms this hypothesis to be true.

*Records having personal author whose surname begins with "SM"

B. Microstructure

1. N-Grams

The most successful microstructure studies in this research project involved n-grams, specifically single letters, two letters (bigrams), and three letters (trigrams). To illustrate the use of n-grams, one can look at the name "JOHNSON." This name has seven letters: J-O-H-N-S-O-N. Five of these letters are distinct, or unique: J-O-H-N-S. The name also contains six overlapping bigrams: JO-OH-HN-NS-SO-ON. In this research, the bigrams were extended to include a blank character (Ø) both before and after each word. Therefore, the example contains eight bigrams: ØJ-JO-OH-HN-NS-ON-NØ. In a like manner, the example contains seven trigrams: ØJO-JOH-OHN-HNS-NSO-SON-ONØ

A simple tabulation of the 41,840 names in the 1% sample generated 338,878 bigrams, including 648 unique bigrams. Since the total number of possible unique bigrams is 729 (27 x 27), the number of unique bigrams generated was 89% of all the possible combinations. This finding is in contrast to findings on English language words where it is widely claimed that only 40% of the possible letter pairs occur at all (RISEMAN). The 1976 sample of 343,593 names contained 691 unique bigrams, or 95% of the possible combinations, of which 690 occurred three or more times.

The same sort of tabulation for trigrams occurring in the 1% sample showed that 7,590 trigrams out of a possible 19,683 trigrams, or 39%, occurred at least once. As with bigrams, this percentage can be expected to increase with larger file sizes.

2. Generation of Random Names

These bigram and trigram counts can be used to generate random names--strings based solely on the frequency with which bigrams and trigrams occur. Generation of random names was first done by SHANNON and offers a feel for the quality of information obtained by such counts that a numerical distribution cannot give.

Random names based on bigram frequencies have very little of a "name" quality about them. However, when trigrams are used, many of the names become plausible (Table 3). It should be noted that no length information has to be used in this generation, other than that contained in the trigram occurrence tables. These names were generated by picking a trigram beginning with a blank in proportion to the frequencies for which such trigrams occur. If, for instance, ØTA was chosen, then the next trigram is based on the frequencies which TAA, TAB,...TAZ, TAØ are likely to occur.

Table 3. Random Names Based on Trigram Frequencies

Mesetz	Li
AØBranne	Lacdo
Rananateidwilveing	Nguegon
Einde	Johmsal
Frivallocq	Do
Pay	HØAdo
Bhargiha	Alineramandaveniedgswam
Himoscoleoggemayd	Oshmatton
Tagefraldman	Samer
Mild	Illose
Keetton	Kri
Henkeley	Sullin
Bux	Wolhoyard
Quincirtz	Clinson
Kov	Sch
Ruzm	Rov
Mozarlmer	Ley
Hain	ØUhner
Catannetzbulteh	Ion
Chighathald	Vakirn
Pallint	Tudson
Ey	Bakenbahaws
Casmichwayogton	Figh
Smiliath	Spethegergoodiconissoldwillore
BØEllerts	Baskir
Bakin	Polber
Man	Ron
Ce	Ston
Sinlesboser	Th
Belson	Berooleyd
Cal	Maby
Gran	Her
Infullbiam	Poe
Ligm	Lin
Tankmou	Neiddons
Bri	Witwoorezips
Ud	Li
Hombrupca	Wal
Barsleames	Pin
Keradarg	Ker
Gres	Palles
Zottman	Narnie
Moussan	Roan
Lann	Robak
Maylompf	Frombickeldel
Gre	Venber
Petter	Ver
Rintammoz	Cart
Zelf	Hofs

It was noted that random names based on trigram frequency are not of higher quality because a common trigram (such as "SMI") is given the same weight when the beginning or the middle of a random name is being generated--that is, the frequency tables have no positional information. Table 4 shows the results of applying positional frequencies to bigrams.

Table 4. Names Generated from Positional Bigram Counts

Stepbeal	Maygrte	Nuirzer	Deary
Tharilmon	Prumorr	Hoetey	Meimash
Rornrord	Neraerd	Yentzmelhimall	Ardi
Martscn	Cryer	Carce	Sottlams
Lenfska	Lochrtz	Brel	Pobarms
Baler	Horinn	Ipsaet	Whahnenilidy
Tokichins	Bust	Ainn	Gelars
Zoberdicolanoghat	Wetershnading	Jareke	Hietin
Freber	Canin	Larking	Bace
Gantin	Gastin	Welasoladerz	Vellourerd
Borg	Drter	Hafmas	Orlukin
Strspons	Beras	Lardeds	Fomith
Shanbell	Lalposh	Pioller	Witnebrill
Gantky	Sckanrner	Mantson	Winnnbemavius
Fourti	Bill	Shanfist	Biaer
Eilus	Flalir	Darad	Fauvan
Dekis	Ilurzy	Sesculey	Ewaro
Zoblerabitreton	Seenheng	Wepleragan	Leulong
Imilay	Halpis	Crice	Bare
Hoynnn	Synthice	Talleracz	Nozeano
Stherill	Warsfrconi	Preergan	Ipsada
Bron	Gosion	Buey	Lachass
Straravam	Silisach	Gohern	Nopleck
Pausten	Masavet	Momaaza	Itrsou
Grarrd	Wakeserlini	Runewove	Boss

Names in Table 4 were first generated by selecting a random length, and then weighted by the actual length distribution of surnames. Using a table of bigram frequencies for each position from the front of the surname, successive bigrams were selected as in the trigram names above. This procedure was followed to the midpoint of the name at which point bigrams were selected based on their positional frequency from the rear of the surname. This required keeping approximately 20 frequencies for each bigram (10 for positional frequency from front and 10 for positional frequency from rear). This is slightly fewer frequencies than required by a simple trigram table, but appears to produce more natural sounding names.

3. Binary N-Grams

In fact, RISEMAN and ULLMAN have proposed that much less information is still useful in error correction; that simply the knowledge of whether a particular bigram or trigram can occur in a given position is enough to accurately detect, and in some case correct, incorrect strings.

To test this theory on names, bigram occurrence checks were done on the 1% sample. These results are presented in Table 5. The bigrams used in this technique are not necessarily adjacent. For example, "JOHNSON" would generate bigrams:

JO, JH, JN, JS, JO, JN
OH ON OS OO ON
HN HS HO HN
NS NO NN
SO SN
ON

for the forward positions, and an equal number for positions relative to the rear position.

Table 5. Number of Bigrams Found for All Position Combinations in the 1% Sample

Unique forward bigram counts:			Unique backward bigram counts:		
1	2	352	1	2	403
1	3	601	1	3	535
1	4	625	1	4	594
1	5	614	1	5	604
1	6	610	1	6	595
1	7	603	1	7	589
1	8	573	1	8	555
1	9	537	1	9	519
1	10	501	1	10	476
2	3	460	2	3	450
2	4	522	2	4	577
2	5	548	2	5	603
2	6	537	2	6	591
2	7	516	2	7	568
2	8	483	2	8	556
2	9	429	2	9	517
2	10	372	2	10	493
3	4	577	3	4	534
3	5	606	3	5	613
3	6	631	3	6	632
3	7	626	3	7	612
3	8	586	3	8	584
3	9	547	3	9	552
3	10	522	3	10	534
4	5	562	4	5	560
4	6	596	4	6	632
4	7	617	4	7	635
4	8	605	4	8	604
4	9	560	4	9	570
4	10	534	4	10	548
5	6	524	5	6	528
5	7	555	5	7	603
5	8	565	5	8	608
5	9	538	5	9	584
5	10	508	5	10	537
6	7	477	6	7	515
6	8	532	6	8	580
6	9	525	6	9	576
6	10	493	6	10	547
7	8	453	7	8	469
7	9	498	7	9	561
7	10	507	7	10	550
8	9	407	8	9	421
8	10	474	8	10	502
9	10	388	9	10	392
	Total:	23,896		Total:	24,808
	72.8% of array filled.			75.6% of array filled.	

As can be seen in Table 5, many positions have nearly as many as the 729 possible bigram combinations. In fact, 73% of the forward possibilities, and 76% of the backward possibilities occurred. Positions 1 to 5 are 83% full. This "density" in the matrix showing possibilities was so high that the figure then was counted for positional trigrams. We found that 17% of the forward positions and 15% of the rear positions had at least one occurrence in the 1% sample of surnames.

Typical results of the application of this information are presented in Table 6.

Table 6. Application of Positional Binary Trigrams

```
HICKEY
++++
  ++++

SANDERS
+++++
  +++++

SAMDERS
+--++
  ++-++

PICONE
++++
  ++++

BUTLER
++++
  ++++

BULTER
++++
  ++++

BUTIER
++++
  ++++
```

The '+' marks indicate that the trigram in that position is an accepted one, a '-' that it is not. Positions are measured to the middle from both the front and rear of the name. In the examples shown, the procedure caught the misspelling of SANDERS as SAMDERS, but did not complain about BULTER for BUTLER. Because of its low discrimination, this approach does not appear to be useful in the control of names.

3. Phonetic Structure

Another approach to name microstructure is based on the phonetic structure of names. Phonetic structure was investigated in the project by applying some known syllabication algorithms for English to names, and by developing an entirely new approach to breaking names into syllables.

The first program was based on the algorithm by RICH. This simple program bases its splits on a small group of special bigrams which are never split, and on the presence of vowels, double consonants, etc. A slightly more complicated algorithm was proposed by MOITRA. MOITRA's algorithm was also implemented in the project and the results are shown along with the RICH algorithm in Table 7.

Table 7. Samples of Hyphenation

RICH	MOITRA
Al-bright	Al-bright
An-cona	Ancona
Arnold	Arnold
Bach	Bach
Bar-banell	Bar-banell
Baugh-man	Baugh-man
Bel-mar	Bel-mar
Ber-thol-dy	Bertholdy
Blake	Blake
Bonne-foy	Bon-nefoy
Brad-ley	Bradley
Brof-ferio	Brof-ferio
Buck-ley	Buck-ley
Byron	Byron
Caring-ton	Caring-ton
Ca-vanah	Cavanah
Chenost	Chenost
Cle-ghorn	Cleghorn
Cot-ton	Cot-ton
Cuen-tas	Cuentas
Da-vid	David
Dell	Deil
Diste-fano	Distefano
Drep-per	Drep-per
Eades	Eades
Elms	Elms
Fa-ber	Faber
Fes-ta	Festa
Fol-liet	Folliet
Free-ling	Freeling
Gale	Gale
Gei-sel	Geisel
Gior-dano	Giordano
Gon-za-lez	Gon-zalez
Gray	Gray
Guedes	Guedes
Hall	Hall
Har-ris	Har-ris
Haz-litt	Haz-litt
Her-shon	Hershon
Hodg-son	Hodg-son
Hoss-ley	Hoss-ley
Hun-ter	Hunter
Jack-son	Jackson

Both of these algorithms have several drawbacks. They miss some obvious breaks, make others they should not, and in general, incorporate very little knowledge about the structure of even the more common names. Their main advantage, however, is that they take very few comparisons for hyphenation and are therefore fast.

The lack of knowledge of the relationships between names in the above two algorithms prompted the development in this project of a method of syllabication similar to that of HAFER. When an alphabetical list of names is shown, some natural breaking places present themselves:

GREEN
GREEN-BAUM
GREEN-BERG
GREEN-BLATT
GREEN-E
GREEN-ER
GREEN-FELD
GREEN-FIELD

Based on this observation, the project team designed its own algorithm that works from both ends of a name. This algorithm reads in the four names which surround the name when the file is alphabetized both forward and backward and then attempts to split the name using as much information from the other names as possible to break the given name into syllables. For example, JACKSON is matched the following way:

Forward Match	Backward Match
JACKA	HICKSON
JACKMAN	DICKSON
(JACKSON)	(JACKSON)
JACOB	HOWISON
JACOBI	WISON

and generates the following splits:

JACK-SON From forward matching
JA-CKSON From backward matching

The process is then repeated recursively on the syllables found so far until no more splits can be made. In this case JA and SON cannot be split any further; however, JACK is matched thus:

JACCOTTET	BLACK
JACINEVICIUS	LACK
(JACK)	(JACK)
JACKA	CHERNIACK
JACKMAN	FLEISCHHACK

and backward matching produces a J-ACK split, which ends the splitting for a final syllabication of J, ACK, JA, CKSON, SON. It should be noted that the breaks produced are overlapping.

The main drawback for large-scale application of this new algorithm is its slow execution time--each split requires several computer disk accesses to a large indexed file of surnames. A second possible problem is that this algorithm tends to break names into very small pieces, e.g., JACK into J-ACK, or UNG into UN-G. Many of these small splits do, however, make linguistic sense when the splitting process is examined in detail.

Unfortunately, both the new and existing syllabication algorithms share the implementation problem of a very large number of different syllables being produced. This problem makes indexing names by their syllables very difficult. Because of this, retrieval experiments could not be completed in the time frame of this grant. In the next section, the more promising techniques of n-grams and clustering for name retrieval are discussed.

V. NAME RETRIEVAL

Two basic techniques were explored for name retrieval. The first relied on breaking the name up as in the microstructure studies, the second is a clustering algorithm based on string distance measures described in Section VI.

A. Microstructure

The most successful experiment was based on bigrams. Bigrams are a prime candidate for microstructure retrieval techniques because there is a manageable number of them (729 in this study), and they preserve some ordering information. For an example of applying this technique, let us use the name BUTLER, misspelled BULTER. This is a very common mistake, as in lowercase the name Bulter will often go unnoticed. The two names break into the following bigrams:

ØB BU UT TL LE ER RØ
ØB BU UL LT TE ER RØ

Each has seven bigrams, four of which are the same. For retrieval, three microstructure approaches are apparent:

(1) Look for names with all of the same letters (anagrams). This works well for transposition errors, but is otherwise limited.
(2) Transpose each bigram in turn, search for an exact match. When variations caused by dropped or added letters are also checked, this results in a large number of searches.
(3) Search for names with at least four of the seven bigrams present.

We have done experiments with techniques 1 and 3. Number 3 seems to offer the only real promise for detecting differences other than the basic typographical ones (transposition, replacement, addition, and omission). In particular, the common variations, "Tchaikovsky" and "Chaikovskii", introduced by different romanization schemes for Cyrillic, should be at least potentially retrievable. Analyzing the bigrams, we have:

ØT TC CH HA AI IK KO OV VS SK KY YØ
ØC CH HA AI IK KO OV VS SK KI II IØ

Both have 12 bigrams with eight of them in common. "TCHAIKOVSKY" has 10 unique characters; "CHAIKOVSKII," eight, all of which are contained in the 10.

Following are the actual search results, trying to identify CHAIKOVSKII as a variant of TCHAIKOVSKY:

Using eight out of 10 letters and a length requirement of 11 ± 2:

CHAIKOVSKII
SHOSTAKOVICH
MACKINTOSH
BARYSHNIKOV
KOSCHATZKY
KRATOCHVIL
MASHKOVICH
SAVOCHKIN
SHAKHNOVICH
STACHOWIAK
STANKOVIC
TOLCHINSKY
TOPACHEVSKYI
VYSOTSKAIA

Using eight out of 12 bigrams:

2	CHAIKOVSKI	3
93	CHAIKOVSKII	4
1	CHAIKOVSKAIA	5

The number in front of each name above is the number of entries found in the 1976 sample of over 300,000 names. The trailing number is a distance measure which will be explained in detail in Section VI. The length mask is very useful when using single letters--in the above example, it cut retrievals from 29 to the 14 displayed.

These searches proceeded by finding all names with any combination of any eight letters, and, in the second case, any combination of any eight bigrams. For the letters, this amounts to 10 things taken eight at a time, or 45 combinations, and for the bigrams 12 things take eight at a time, or 495 combinations. The 495 combinations is a very large number and fortunately approximates what might be considered the worst case one could expect to encounter for any name.

For reasonable execution speeds and storage requirements, the file of surnames is inverted into 729 records, one for each possible bigram. Each record in this file is in fact a "bit vector", each bit of which indicates whether the corresponding name contained that bigram (see LEFKOVITZ). As each name is searched, its bigrams are extracted, duplicates eliminated, and the corresponding bit vectors read from the inverted file into memory. To support a file the size of the 1976 sample, with over 109,000 unique surnames, this requires a substantial amount of computer memory (13 K bytes per bigram).

Each combination satisfying the search is then generated by a combinations algorithm (see REINGOLD, pp. 179-181) and the bit vectors ANDed together to find all names with those bigrams. This result is then ORed with previous combination results and the process continued. The main drawbacks to this technique are:

(1) Large core requirements for large files,
(2) Large number of combinations required.

Searches on our Sigma-9 computer take a substantial amount of CPU time on the large (109,000 unique names) 1976 file--from 5-10 seconds to 1-2 minutes. Substantial increases in speed might be possible if the ANDing and ORing of bit vectors were implemented in microcode, or a procedure for eliminating many of the recurring combinations which are now done could be developed.

Searching for all names with all, or all but one, of the same characters is especially useful when checking for transposition and omission errors. The following are misspellings which have occurred for the name HSIAO: SHAIO, SHIAO, SHAO, HSAIO and HSIAD. This is a case where the majority of the bigrams are affected by the error, but search for names with all but one letter will retrieve the correct name.

To eliminate names which contain similar characters but are much longer than the input name, a length mask was implemented, also with bit vectors, to allow names within a specified length range to be quickly selected. This length masking was so successful with the single-character search that it was also implemented in the bigram search and is reflected in the examples given in Section VI.

When the single-character search is tried with the misspelled name "BULTER" looking for names with all six letters, 45 names were retrieved from the 1% sample. When screened for a length of six, two names remained: BUTLER and BURTLE. If the length was allowed to vary by one, five records were retrieved: BUTLER, HULBERT, BOULTER, BURTLE and TRUMBLE. The limitations of this technique become clear, however when the matching is slightly relaxed; a search on any five of six characters, allowing the length to vary by one, retrieves 123 surnames. A similar search on "SANDERS" retrieved only 30 records, but many of these are not even close, such as BESNARD, ERDNASE, and MARSDEN.

B. Clustering

HALL suggests the possibility of clustering similar names in order to facilitate retrieval. In this method a tree is formed. For example, clustering the most frequent 20 names in the SM sample gives the results shown in Figure 7.

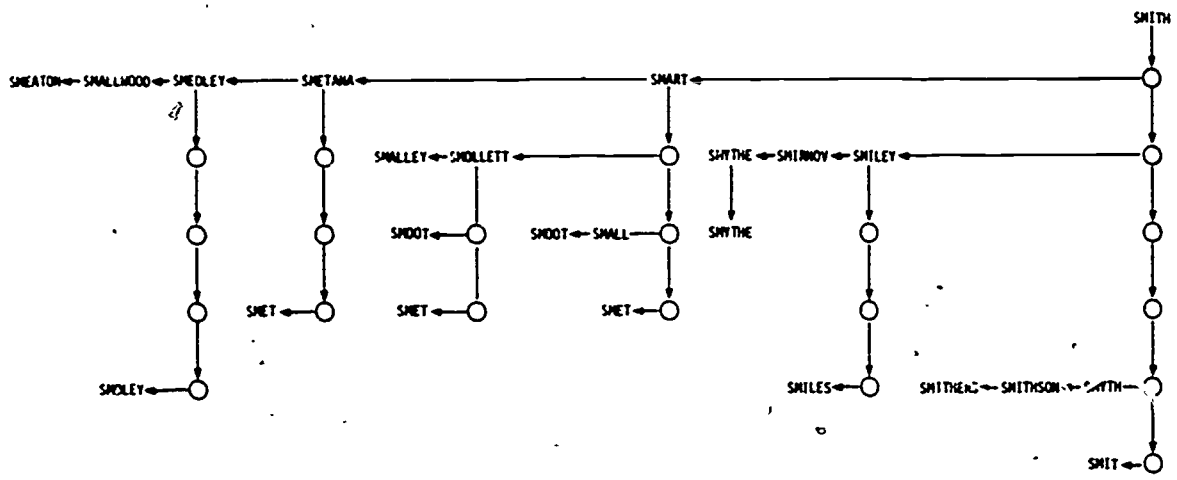

Figure 7. Sample Clustering of the Most Frequent 20 Names in the SM Sample

An important aspect of this clustering technique is that, although it is not order independent (i.e., the order in which names are entered into the tree affects the tree shape), there is a natural order to use--most frequent first. This method guarantees that such common names as SMITH, BROWN, JOHNSON, etc., will head major sections of the tree. This seems to be exactly what is needed to force common variations and misspellings to cluster under the predominant variation.

To cluster names, some sort of distance measure between names is needed. A number of such measures are possible (see ALBERGA). HALL suggested using the Damerau-Levenshtein metric (see DAMERAU) which, as its name implies, has the advantage of being a "metric." This means that the familiar triangle inequality of plane geometry holds with names; if name A has a distance x to B and B a distance y to C and the z is the distance from A to C, then $x + y \geq z$. Unfortunately, this algorithm does not work well with the common name variation of multiple surnames in a different order, since it depends on reducing one string to a transformation of the other using single omissions, insertions, and reversals. For this reason we developed a measure based on a file comparison technique invented by HECKEL.
This algorithm proceeds by finding similarities between names. First, characters which occur uniquely in each are matched, then characters immediately adjacent to these are paired, and the process continues until all characters are either matched or not.

From the tables containing the information on the match boundaries a symmetrical distance measure is derived which reflects the number of discontinuities and replacements needed to transform one name into the other.

The clustering program was first written in FORTH and maintained its clusters on disk. This proved much too slow, so the clustering procedure was recoded in PASCAL, maintaining its clusters entirely in core. This did restrict the number of names which could be handled at one time to less than 1,000 because of core limitations, but this is large enough to handle the entire file of 779 SM surnames. Computation time to build the clusters still seemed rather excessive, at over 13 minutes CPU time, so a simpler comparison algorithm was developed.

This algorithm is based on the number of bigrams that two names have in common, normalized by the mean length of the two names. This is much simpler computationally than the previous algorithm and cut run time to build the cluster nearly in half, while reducing core requirements.

The clustering program makes use of a distance measure between names as the criterion for constructing a structure of records. Each record contains a field for a down pointer and a left pointer, referred to as DLINK and LLINK, respectively. The records also contain either a name field or a null field; the reason for the null field alternative will become evident.

The size and ordering of the cluster depends upon the number of levels allowed and the minimum distance measure value assigned to each level. In the chart depicting the cluster (Figure 7), the levels are the horizontal rows. Although two names appear in the same level, they are related only if there is a LLINK pointer from one to the other. Thus, it is the purpose of the LLINK to link together names which have a distance measure within the range assigned to that level. The DLINK is used to move from a higher level to a lower one and is never used to link two different names. For example, if NAME A was shown to be related to NAME B by the distance measure assigned to level 10, but NAME B appeared in level 11, then a DLINK would be created from NAME B to a newly created record in level 10 and that new record would have a LLINK to NAME A. The new record created in this case would not have a name field but a null field because we know the name associated with it is in the name field of record NAME B. This eliminates redundancy and saves memory.

The names in the upper levels have the least amount of similarity--or even no similarity. As a new name is input for comparison, it is compared to these names, moving right to left following the LLINKs. If the input name shows a higher degree of similarity with a name in the cluster than the level required for the level being scanned, then DLINKs are created, if necessary, to LLINK the input name at the appropriate level. Comparison with other higher level names is then resumed. Once the cluster has been traversed so that the name has been correctly placed, another name can be input and the process begins anew.

After all names have been clustered, the internal list structure is converted to a parenthetical notation for lists used by the LISP programming language. Figure 8 translates the data from Figure 7 into this format. From this, the internal list structure can be recreated easily for searching.

```
(
  (SMITH
    (SMITH
      (SMITH
        (SMITH (SMITH (SMITH (SMITH )(SMIT ))(SMYTH )(SMITHSON )(SMITHERS ))))
        (SMILEY (SMILEY (SMILEY (SMILEY )(SMILES ))))(SMIRNOV )(SMYTHE (SMYTHE ))
)
    (SMART (SMART (SMART (SMART )(SMET ))(SMALL )(SMOOT ))
      (SMOLLETT (SMOLLETT (SMOLLETT )(SMET ))(SMOOT ))(SMALLEY ))
    (SMETANA (SMETANA (SMETANA (SMETANA )(SMET ))))
    (SMEDLEY (SMEDLEY (SMEDLEY (SMEDLEY (SMEDLEY )(SMOLEY ))))(SMALLWOOD )
    (SMEATON )))
```

Figure 8. Output Format for Clusters

Both algorithms were run against the file presented in Appendix A.1: Surname Spelling Errors. For this test, the correct forms of the names were entered into the file and clustered along with the most frequently occurring 1000 surnames.

Table 8 is an example of searching for the name "ANDREWS," using a typographical error omission, "ANDRES." Clustering did rather well in identifying similar names; the original algorithm found 23 to 26 of the 30 test variants, depending on the number of cluster levels. The second algorithm based on bigram comparisons found all 30. The number of comparisons needed for this performance was excessive, ranging from 25% to 75% of the total file. This table also shows the results of searching with bigram combinations for comparison. The level 1 matches for each algorithm correspond to the closest matches found.

Table 8. Typical Search Results

	MATCHES			
	Level 1	Level 2	Level 3	Number of Comparisons
Heckel algorithm Clustering	ANDREWS	BRENNAN FREEMAN REEVES BANKS EVANS	LANDAU REMY HANDEL GREY REED WEST	414
Common bigram	ANDREWS	ANDRADE ANDERSEN ANDERSON		811
Bigram combinations	ANDREWS	ANDREWS ANDRE ANDREAS ANDRESS ANDRIES ANDERES ANDREGS BANDRES	ANDREW ENDRES ANDREE LANDES ANDREI ANDREN ANDRUS LANDRE RANDLES ANDRAS ANDREA	68 + bit vector indexes

VI. NAME PROBABILITIES

A. Name Ranking

By combining one of the name similarity measures described in Section V-B with the bigram combinatorial search (Section V-A), a system is obtained capable of retrieving surnames and ranking them much like the clustered search does. Table 9 gives examples of typical searches done using this system, run against the 1976 data base of 109,000 unique names. The numbers to the left of each name indicate the frequency of occurrence in the file; the numbers to the right, the distance as measured by the HECKEL algorithm used to rank them. The desired name is boxed if the program was successful in finding it. Direct comparisons with the clustering algorithm are not possible because of the differences in data bases used. It should be noted that the desired name is not necessarily contained in the file being searched.

Table 9. Ranked Bigram Combination Searches

Please type surname: KESSERLING
5 Bigrams out of 8 used for searching.
56 Combinations to be generated.
107 Records found before length masking.
69 Records after length selection.

2 KEYSERLING	2
2 KESSELRING	3
1 KEIDERLING	4
1 KEMMERLING	4
5 KERSTING	5
3 KAMERLING	5
1 KIMERLING	5
27 STERLING	6
7 SPERLING	6
3 KLINGBERG	6
2 BEERLING	6
2 KIMBERLING	6
2 VIERLING	6
2 WOELKERLING	6
1 EVERLING	6
1 KEESLING	6
1 KEPPLINGER	6
1 KOBBERLING	6
1 WAKELING	6
1 ZINSERLING	6

13.316 Seconds cpu time for bv matching.
3.986 Seconds cpu time for top 20 selection.

Please type surname: KIPPIN
4 Bigrams out of 7 used for searching.
35 Combinations to be generated.
133 Records found before length masking.
57 Records after length selection.

2 PIPPIN	2
1 KOPPIN	2
14 KIPLING	3
1 KIOPINI	3
1 KIPPHAN	3
1 KOPIN	3
1 PHIPPIN	3
1 PIPPING	3
1 TIPPING	3
3 COPPIN	4
3 KARPIN	4
3 KIPPAX	4
3 KISSIN	4
2 KIRWIN	4
2 LAPPIN	4
1 KEMPIN	4
1 KILLIN	4
1 KIPPEL	4
1 KIPPES	4
1 LOPPIN	4

7.312 Seconds cpu time for bv matching.
2.830 Seconds cpu time for top 20 selection.

Please type surname: HEWLITT
5 Bigrams out of 8 used for searching.
56 Combinations to be generated.
5 Records found before length masking.
4 Records after length selection.

36 HEWITT	1
7 HEWLETT	2
9 HEWETT	3
7 HAZLITT	4

13.276 Seconds cpu time for bv matching.
0.656 Second cpu time for top 20 selection.

Please type surname: SHAUGNESSY
5 Bigrams out of 8 used for searching.
56 Combinations to be generated.
7 Records found before length masking.
5 Records after length selection.

10 SHAUGHNESSY	1
4 OSHAUGHNESSY	2
2 SHAINESS	4
8 HENNESSY	6
3 SHARPLESS	7

13.092 Seconds cpu time for bv matching.
0.732 Second cpu time for top 20 selection.

This approach gave excellent results, although it can be rather slow when large numbers combinations are generated. If combined with the next stage of name-matching as described in the following section, it seems possible that a useful author-matching method would result that exceeds present ones.

B. Full-Name Comparison

After candidate surnames have been identified, there remains the problem of comparing full names. We developed a fairly sophisticated program to test the application of a matching methodology proposed by the principal investigator of this project (HICKEY). The basis of this technique is to set up a decision table tabulating the possible ways which names can match. In this table "E" means that an exact match is required; "P", that a partial match or better is needed; and "N" that no match is required. Each column of Table 10 represents a set of criteria for an acceptable name.

Table 10. Name-Matching Decision Table

Forename	P	E	P	E	N
Middlename	P	P	E	N	E
Surname	P	E	E	E	E
Date	E	P	P	E	E

For example, the second column specifies that if two names match exactly on their forenames and surnames, and have partial matches on their middle names and dates, they are considered an overall match. Each of the rows may have different criteria for what is considered an E, P, or N match.

The forenames and middle names employ the same criteria: the match is exact if they are the same, and more than an initial is present; partial if both have the same initial, or one or both are not present; and no match otherwise. The surnames must be the same for an exact match. All other cases are considered no match; the assumption being that alternative surnames have been identified through some other process.

The date-matching is slightly more complicated. The birth and death dates are first matched individually. If both are present and the same, there is an exact match; if both are present and different, there is a no match. If either is missing, it is a partial match for that date. The overall date field is considered an exact match if the birthdate is an exact match and the death date is either exact or partial. If the birthdate is a partial match, and the death date is partial or better, the date field is an overall-partial. Otherwise, the date field is a no match.

The following are examples of this matching, each corresponding to the minimum information needed to satisfy a corresponding column in Table 10:

Column	
1	J. Smyth, [redacted]1 vs. J. Smith, 1901
2	John Smith vs. John Smith
3	J. Paul Smith vs. John Paul Smith
4	John Q. Smith, 1901 vs. John R. Smith, 1901
5	James Paul Smith, 1901 vs. John Paul Smith, 1901.

The example for column one assumes that a partial match surname has been identified outside the matching process being described here. It should be noted that an actual system could make use of clues outside the personal name field to improve judgements as to whether two names are the same. Another limitation of this algorithm is that it ignores the frequency of occurrence of names, information important in the human matching of names.

The matching described above takes place between an input name and all names in the file being searched, which have any possibility of satisfying the given criteria. To accomplish this, the file to be searched is set up with an index consisting of a 15-byte key. The key consists of the first seven characters of the surname, the first three characters of the forename, the middle initial, and a 4-byte field to ensure uniqueness of the key. The records then are retrievable by any contiguous portion of the key, starting with the surname. These records can be accessed sequentially after a retrieval. If the input name has no forename, or only an initial, one key is generated using that information. If a full forename and no middle initial is present, or first and middle initials, then two keys need to be generated. Three keys are needed if both middle and forename are present in the input name. This ensures that the searching program can begin its search at each of these points in the file and proceed matching until an unequal name portion of the key is found, and still examine all possible candidate records. For the three-key situations, one key would be built simply from the first initial and surname; the second from forename and surname; the third from forename, middle initial, and surname.

Table 11 gives some representative examples of this search as implemented on the SM file. It should be noted that this matching is on what may be the most difficult section of a file of names, and is using a complete subsection of a data base of 6 million names. Even with the most common names, a display could summarize the different authors found, in less than 100 lines of output. A typical name would easily fit on a single CRT screen.

Table 11. Example of Full-Name Search Implemented on the SM File

		Number of Entries
Name input:	Smith, M. Brewster	
Matches:	Smith, Mortimer Brewster, 1906	(12)
	" Mortimer Brewster, Ed.	(1)
	" Mortimer Brewster, 1906 Ed.	(1)
	" Mahlon Brewster, 1919	(10)
Name input:	Smith, M. A.	
Matches:	Smith, Merrill A.	(1)
	" Melody A.	(1)
	" Maxwell A., Ed.	(1)
	" Martin A.	(2)
	" Marina A.	(1)
	" Marilyn A.	(1)
	" Marc A.	(2)
	" Macon A.	(1)
	" M. A.	(2)
Name input:	Smith, M. E.	
Matches:	Smith, Morton E.	(1)
	" Milton E.	(1)
	" Meredith E.	(2)
	" Melden E.	(1)
	" Maurice E.	(1)
	" Matthew E.	(1)
	" Marvin E.	(2)
	" Mark E	(1)
	" Marck E., 1934	(1)
	" Marjorie E., 1900	(1)
	" Maria E.	(1)
	" Margaret E.	(1)
	" Mabel E.	(1)
	" M Estellie	(1)
	" M Estellie, 1935	(5)
	" M Elizabeth	(2)
	" M. Eileen	(1)
	" M E	(2)

VII. SUMMARY

A. Discussion

Within the grant, a great deal of basic work on the microstructure of surnames was completed and some progress was made on utilizing that knowledge in searching very large data bases. The data bases of interest to libraries are indeed very large and give every indication of continued growth, both in number of unique surnames found and of course, in the number of different authors represented. It seems that the number of different author surnames available soon will surpass one million, creating a very densely populated "microspace" when the average length is only seven characters.

The techniques developed and tested within the grant could have only a limited application in operational data base systems with millions of records. One such application might be to search incoming names against a file of 10,000 to 100,000 of the most common surnames. If this comparison turned up a probable match, then an extended matching against names with that surname could be performed. The extended matching would almost certainly eliminate misspellings of "Shakespeare," for example, and would resolve variant forms of the more common authors (such as "Tchaikovsky" versus "Chaikovskii"). A check could be run "on the fly" in a system such as OCLC's every time a new record was added to the data base. Such a check could also be run on existing records, printing out error reports that then would be manually checked and modified in the data base.

Even such a limited application would require a significant overhead for such verification/correction. It is conceivable that 100,000 surnames could be kept in core memory in a computer and rapidly searched by a clustered search of bigrams. However, full names, dates, and other relevant information needed for matching are so numerous that, with current technology, this information must be kept in disk storage. Therefore, a fairly high processing overhead is imposed.

The ideal system would make even greater demands on computer resources. Each author name entered would be conceptually matched against the entire file of names for complete control. To be efficient, this system would require hardware of different design than a general-purpose computer.

B. Further Research

To continue the research presented here, a number of problems need to be addressed:

(1) better estimates of the universe of surnames,
(2) forenames studies,
(3) optimized matching algorithms for different languages with a method of identification,
(4) applications of specialized hardware, such as the use of optical computers for similarity matching,
(5) more investigation of clustering algorithms for names and strings in general, and
(6) better differentiation of names on the basis of pronunication.

C. The Future

A bright future reveals itself for the directions started in this research, although large-scale application is some time away. In general, however, automated cataloging systems will move inevitably into more "intelligent" systems--both the human interface, and data base organization will become more useful through the integration of added knowledge about author names and artificial intelligence techniques. This integration will occur through continued development of the cataloging systems, and/or the adoption and application of "intelligent" data base systems developed in other areas.

APPENDIX A: THIRTY NAME VARIANTS*

A.1. Surname Spelling Errors

TEXT FROM RECORD:	REQUESTED CHANGE:
Andraa, Edgar Harold	Andrews, Edgar Harold
Avalle-Acre, Juan Bautiata, ♦e ed.	Avalle-Arce, Juan Bautiata, ♦e ed.
Azavedo, W. Vernon	Azevedo, W. Vernon
Beiderweiden, George	Beiderwieden, George
Burrows, Thomae	Burrowes, Thomae
Buecaglie, Lao F.	Buecaglia, Lao F.
Chayefsy, Paddy, ♦d 1923-	Chayefsky, Paddy, ♦d 1923-
Dairymple, Dana G.	Dalrymple, Dana G.
Del Ray, Leetar, ♦d 1915-	Del Rey, Lester, ♦d 1915-
Diubailo, P. K.	Dziubaila, Pavel Kuz'mich
Durkheim-Montmartin, Karlfried	Durckheim-Montmartin, Karlfried
Fuch, Gunter, ♦d 1920-	Fuchs, Gunter, ♦d 1920-
Goldman, Richard M.	Goldman, Richard Martin, ♦d 1931- ♦e joint author.
Gonsalus Gónz'alee, Catherine.	Gunsalus Gonz'alez, Catherine.
Henshel, Stan.	Henschel, Stan.
Hernadez, Joe'e ♦d 1834-1886. ♦t Martin Fierro.	Hernandez, Jose', ♦d 1834-1886. ♦t Martin Fierro.
Hewlitt, Maurice Henry, ♦d 1861-1923.	Hewlett, Maurice, Henry, ♦d 1861-1923.
Jerret, Bede, ♦d 1881-1934	Jerrett, Bede, ♦d 1881-1934
Kesserling, Joseph, ♦d 1902-1967.	Keeselring, Joseph.
Kippin, Andrew, ♦d 1725-1795.	Kippie, Andrew, ♦d 1725-1795.
Licke, William John, ♦d 1863-1930.	Locke, William John, ♦d 1863-1930.
McClein, Mary Webetar.	McLain, Mary Webster
Rubinatein, Murray.	Rubenstein, Murray.
Saint Beuve, Charles Auguetin, ♦d 1804-1869.	Sainte Beuve, Charles Auguatin, ♦d 1804-1869.
Sheugneesy, Mary Ellen, ♦d 1938-	Shaughneasy, Mary E.
Velaquez, Diago Rodriguez de ...	Velazquez, Diego Rodriguez de ...
Vickke, Bernard Hubertue Maria	Vlekke, Bernard Hubertue Maria
Yorkal, Felix, Y.	Yokel, Felix Y.
Zeigler, Barnhard	Ziegler, Bernhard
Ziedman, Irving	Zeidman, Irving

*Names in Appendix A extracted from error reports corrected by OCLC's Bibliographic Maintenance Section.

A.2. Forename Spelling Errors

TEXT FROM RECORD:	REQUESTED CHANGE:
Ball, Mary Margaret, ♦d 1909-	Ball, Mary Margaret, ♦d 1909-
Beck, Herbert F.	Beck, Hubert F.
Bonk, kWallace John, ♦d 1923-	Bonk, Wallace John, ♦d 1923-
Booth, Henry Kendel	Booth, Henry Kendall
Castro Leal, Atonio, ♦d 1895- ♦e ed.	Castro Leal, Antonio, ♦d 1895- ♦e ed.
Daniele, Babe Virinie, ♦d 1901- ♦e comp.	Daniele, Babe Virginia, ♦d 1901- ♦e comp.
Easton, ALice	Easton, Alice
Evens, Christopher Richie	Evans, Christopher Riche
Grimm, Jakob Lubwig Karl, ♦d 1785-1863. ♦t Rumpleetilzchen.	Grimm, Jakob Ludwig Karl, ♦d 1785-1863. ♦t Rumpleetilzchen.
Hubetsch, Walter	Hubatsch, Walther
Kierkegaard, Soren Aabye, ♦d 1813-1855	Kierkegaard, Séren Aabye, ♦d 1813-1855
Klaiber, Christian Benjamin, ♦d 1795-1836	Klaiber, Christoph Banjamin, ♦d 1795-1836
Notovich, Nickolai, ♦d 1858-	Notovich, Nikolai, ♦d 1858-
Raymond, Irvin Woodworth, ♦d 1898-	Raymond, Irving Woodworth, ♦d 1898-
Serova, Svetland Andreevna.	Serova, Svetlana Andreevna.
Shakespeare, Willia, ♦d 1564-1616.	Shakespeare, William, ♦d 1564-1616.
Spicq, Celaus, ♦d 1901-	Spicq, Ceslaus, ♦d 1901-
Strelka, Josef	Strelka, Joseph, ♦d 1927-
Stroud, Nikolas	Stroud, Nicholas
Swinnerton, Frank Author	Swinnerton, Frank Arthur
Thibault, Pierette	Thibault, Pierrette
Tinker, Edward Laroque, ♦d 1881-	Tinker, Edward Larocque, ♦d 1881-1968.
Van Dyke, Venda Kay	Van Dyke, Vonda Kay
Vivaldi, Antonia, ♦d 1678-1741	Vivaldi, Antonio, ♦d 1678-1741
Woodward, Geroge Evereton.	Woodward, George Evereton.
Yeats, William Butlen	Yeats, William Butler

A.3. Dates, Punctuation

TEXT FROM RECORD:	REQUESTED CHANGE:
Bach, Johann Sabastian, ǂd 1865-1750.	Bach, Johann Sebastian, ǂd 1685-1750.
Bailey, Lloyd R.	Bailey, Lloyd R., ǂd 1936-
Beauvior, Simone de, ǂd 1903-	Beauvior, Simone de, ǂd 1908-
Bryan,William Frank ǂd 1879-	Bryan, William Frank, ǂd 1879-
Bultmann, Rudolf Karl, ǂd 1844-	Bultmann, Rudolf Karl, ǂd 1884-
Cartland, Barbara.	Cartland, Barbara, ǂd 1902-
Chen, Chi-hau	Chen, Chi-hau ǂd 1937-
Chirico, Giorgio de, ǂd 1888-	Chirico, Giorgio de, ǂd 1888-
Cordeiro, Jose Pedro Keita, ǂb 1914-	Cordeiro, Jose Padro Keita, ǂd 1914-
Davis, Harbert John, ǂd 1983-,	Davis, Harbert John, ǂd 1883-1967.
Faulkner, Virginia.	Faulkner, Virginia, ǂd 1913-
Gaijeal. J. M.	Gaijeal, J. M.
Hanson, Earl Parker, ǂd 1809- ǂe ed.	Hanson, Earl Parker, ǂd 1899- ǂe ed.
Harrison, Benjamin, ǂc Pres. U.S., ǂd 1863-1901.	Harrison, Benjamin, c Pres. U.S., ǂd 1833-1901.
Hofmann, Liesslotte, ǂd 1914- ǂe joint author.	Hofmann, Liesslotte, ǂe joint author.
Husa, Karel, ǂd 1915-	Husa, Karel, ǂd 1921-
Jones, Tom, ǂc librattist. ǂt The fantasticks.	Jones, Tom, ǂd 1928- ǂt Fantasticks.
Kaluzhskaia, Tamara Grigor' evra	Kaluzhskaia, Tamara Grigor'evra
Koch, Harry Walter ǂd 1909-	Koch, Harry Walter, ǂd 1909-
Menendez Pidal, Ramon, ǂd 1869-	Menendez Pidal, Ramon, ǂd 1869-1968.
Milhaud Darius, ǂd 1892-1975	Milhaud Darius, ǂd 1892-1974
Miller,Gordon Wayne, ǂd 1938-	Miller, Gordon Wayne, ǂd 1938-
Neve, Felipe de, ǂd 1740 (ca.)-1784.	Neve, Felipe de, ǂd 1740 (ca.)-1794.
QAmr ibn Barhr, Abau QUthmaan, al-Jaarhirz, ǂd 779?-867?	al-Jahiz, Amr ibn Bahr, ǂd d. 868 or 9.
Osborne, John, ǂc dramatist	Osborne, John, ǂd 1929-
Phillips, Heidi S	Phillips, Heidi S.
Ramsey, Charles George, ǂd 1884-	Ramsey, Charles George, ǂd 1884-1963.
Ramsey, CharlesGeorge, ǂd 1884-	Ramsey, Charles George, ǂd 1884-1963.
Shirk, ǂb John C.	Shirk, John C.
Sleeper, Ha rold Reeve, ǂd 1893- ǂe joint author.	Sleeper, Harold Reeve, ǂd 1893-196G, ǂe joint author.
Sleeper, Harold Reeve, ǂd 1803-	Sleeper, Harold Reeve, ǂd 1893-1960.
Van Every,Dale, ǂd 1896-	Van Every, Dale, ǂd 1896-
Vivaldi, Antonio, ǂd 1680 (ca.)-1741.	Vivaldi, Antonio, ǂd 1678-1741.

A.4. Other Changes to Names

TEXT FROM RECORD:	REQUESTED CHANGE:
Albrecht, Johann Friedrich	Agricola, Johann Friedrich
Arriegade Herrera, Genaro	Arriegada, Genaro, ǂd 1943-
Bell, Thomas M.	Bell, Thomas Matthew
Bessinger, J B	Bessinger, Jess B.
Birke J B	Birke, John Betteley
Birke, J B	Birke, John Betteley
Birke J B	Birke, John Betteley, ǂe ed.
Crawford, S. M. C.	Crawford, G. M. C. ǂq [Gerald Norman Cullen]
Dupre', Marcel, ǂd	Dupre', Marcel, ǂd
Jeanneret, Francios Charles Archille	Jeanneret, Francois Charles Archille
Modenov, P S	Modenov, Petr Sergeevich
Modenov, P. S.	Modenov, Petr Sergeevich
Moore, A. Milton, ǂt Financing Canadian federation	Moore, Albert Milton, 1918- ǂt Financing Canadian federation
Oehlenschlager, Adam, ǂd 1779-1850.	Oehlenschlager, Adam Gottlob, ǂd 1779-1850.
Pollock, Ted.	Pollock, Theodore Marvin, ǂd 1929-
Scott, Walter, ǂc bart., ǂd	Scott, Walter, ǂc Sir, bart., ǂd
Sealey, Leonard George William	Sealey, L. G. W.
Sherrington, Sir Charles Scott	Sherrington, Charles Scott, ǂc Sir,
Weber, Carl J.	Weber, Carl Jefferson, ǂd 1894-1966.
Weiner, John W	Weiner, John M.

APPENDIX B: A PORTION OF THE SM CLUSTER

```
C RUN1PC
(
  (SMITH
    (SMITH
      (SMITH
        (SMITH
          (SMITH
            (SMITH
              (SMITH (SMITH (SMITH (SMITH )(SMITH_Y_SMITH ))(SMITH )(SMITH )))
              (SMITH_V ))(SMIT )(SMITHE (SMITHE (SMITHE )))(SMITCH )(SMAITH )
            (SMIDTH )(SMITH_J_C )(SMITH_J_L )(SMITH_SEBA )(SMITH_URIAH ))
          (SMYTH (SMYTH ))(SMITHSON (SMITHSON (SMITHSON )))
          (SMITHERS (SMITHERS (SMITHERS ))(SMITHER (SMITHER ))
            (SMITHE (SMITHE (SMITHE )))(SMITHES ))(SMITS (SMITS ))
          (SMITHIES (SMITHIES (SMITHIES ))(SMITHES ))
          (SMITHDAS (SMITHDAS (SMITHDAS )))(SMITT (SMITT ))
          (SMITHEY (SMITHEY (SMITHEY (SMITHEY )))(SMITHES )(SMITHEE )
            (SMITHEN ))(SMATH (SMATH )(SMAITH ))(SMITH_GARRY )
          (SMITH_JANIE_S_ )(SMITH_APRIL )(SMITH_BERTHA_H )(SMITH_FAMILY )
          (SMITH_JOEL )(SMITH_KOGAN )(SMITH_MAURY )(SMITH_R_D_E )
          (SMITH_ROY_C )(SMITHNER )(SMITHSI )(SMITHUIS ))
        (SMITHELLS (SMITHELLS (SMITHELLS )(SMITHES ))
          (SMITHEY (SMITHEY (SMITHEY (SMITHEY )))(SMITHES )(SMITHEE )
            (SMITHEN )))(SMITHCORS (SMITHCORS (SMITHCORS )))
        (SMITHERMAN (SMITHERMAN (SMITHERMAN ))(SMITHYMAN (SMITHYMAN )))
        (SMITHLINE (SMITHLINE (SMITHLINE )))
        (SMITTLE (SMITTLE (SMITTLE )(SMITLEY ))(SMITT (SMITT )(SMITTI )))
        (SMITHWICK (SMITHWICK (SMITHWICK ))(SMITHBACK ))
        (SMITTER (SMITTER (SMITTER )(SMITTEN ))(SMITT (SMITT )(SMITTI ))
          (SMITHNER ))(SMITHVANIZ (SMITHVANIZ (SMITHVANIZ )))
        (SMITAL (SMITAL )(SMIL )(SMITHGALL ))
        (SMITH_MONZON (SMITH_MONZON (SMITH_MONZON (SMITH_MONZON ))))
        (SMITHKEARY (SMITHKEARY )(SMITH_GARRY ))
        (SMITHMEYER (SMITHMEYER )(SMITHNER ))(SMITHROSE (SMITHROSE ))(SMIM )
        (SMITHBERG (SMITHBERG (SMITHBERG )(SMITHBURG ))(SMITH_HOLBERG ))
        (SMITHERAM (SMITHERAM (SMITHERAM )(SMITHAM )(SMITHERUM )))
        (SMITHLOVIN )(SMITHSTARK )(SMIECH )(SMITH_ANTHONY_ )(SMITH_BRINDLE )
        (SMITH_DORRIEN )(SMITH_F_KELLY (SMITH_F_KELLY )(SMITH_FAMILY ))
        (SMITH_GEORGE )(SMITH_H_JAY_EX )(SMITH_I_H_LAND )(SMITH_PARKER )
        (SMITHBURN (SMITHBURN ))(SMITHFIELD )(SMITHGOLD )(SMITHHINDS )
        (SMITHURST ))
```

BIBLIOGRAPHY

Alberga, Cyril N. String similarity and misspellings. Communications of the ACM. 10(5): 302-313; 1967 May.

Batori, Istvan. Error detection--linguist's view. Heidelberg Germany: IBM Germany Heidelberg Scientific Center: 1975 April 15; Technical Report TR75.08.006.

Becker, Peter W. Recognition of patterns using the frequencies of occurrence of binary words. New York: Springer Verlag; 1978.

Blair, Charles R. A program for correcting spelling errors. Information and Control, 3: 60-67; 1960.

Carlson, Gary. Techniques for replacing characters that are garbled on input. Proceedings of the 1966 Spring Joint Computer Conference. AFIPS Conference Proceedings 28. 189-192.

Damerau, F.J. A technique for computer detection and correction of spelling errors. Communications of the ACM. 7(3): 171-176; 1964 March.

DeHeer, T. Experiments with syntactic traces in information retrieval. Information Storage and Retrieval. 10: 133-144; 1974 March/April.

Fokker, Dirk W.; Lynch, Michael F. Application of the variety-generator approach to searches of personal names in bibliographic data bases-Part 1. Microstructure of personal authors' names. Journal of Library Automation. 7(2): 105-118; 1974 June.

Galli, E.J.; Yamada, H. An automatic dictionary and the verification of machine-readable text. IBM Systems Journal. 6(3): 192-207; 1967.

Hafer, Margaret; Weiss, Stephen F. Word segmentation by letter successor varieties. Information Storage and Retrieval. 10: 371-385; 1974 November/December.

Hall, Patrick A.V.; Dowling, Geoff R. Approximate string matching. ACM Computing Surveys. 12(4): 381-402; 1980 December.

Heckel, Paul. A technique for isolating differences between files. Communications of the ACM. 21(4): 264-268; 1978 April.

Henderson, Peter. Functional programming, application and implementation. Englewood Cliffs, NJ: Prentice-Hall; 1980.

Hickey, Thomas B.; Rypka David J. Automatic detection of duplicate monographic records. Journal of Library Automation. 12(2): 125-142; 1979.

King, Donald R. The binary vector as the basis of an inverted index file. Journal of Library Automation. 7(4): 307-314; 1974 December.

Lee, R.C.T.; Slagle, James R.; Mong, C.T. Towards automatic editing of records. IEEE Transactions on Software Engineering. SE-4(5): 441-448; 1978 September.

Lefkovitz, David. The large data base file structure dilemma. Journal of Chemical Information and Computer Sciences. 15(1): 14-19; 1975 February.

Lipetz, Ben-Ami; Taylor, Kathryn F. Performance of Rueckings word-compression method when applied to machine retrieval from a library catalog. Journal of Library Automation. 2(4): 266-271; 1969 December.

Lowrance, Roy; Wagner, Robert A. An extension of the string-to-string correction problem. Journal of the ACM. 22(2): 177-183; 1975 April.

McElwain, Constance K.; Evens, Martha B. The degarbler--a program for correcting machine-read morse code. Information and Control. 5: 368-384; 1962.

Moitra, Abha; Mudur, S.P.; Narwekar, A.W. Design and analysis of a hyphenation procedure. Software--Practice and Experience. 9: 325-337; 1979.

Muth, Frank E., Jr.; Tharp, Alan L. Correcting human error in alphanumeric terminal input. Information Processing and Management. 13: 329-337; 1977.

Newcombe, Howard B.; Kennedy, James M. Record linkage making maximum use of the discriminating power of identifying information. Communications of the ACM. 5(11): 563-566; 1962 November.

Nugent, William R. Compression word coding techniques for information retrieval. Journal of Library Automation. 1(4): 250-260; 1968 December.

Peterson, James L. Computer programs for detecting and correcting spelling errors. Communications of the ACM. 23(12): 676-687; 1980 December.

Reingold, Edward M.; Nievergelt, Jung; Deo, Narsingh. Combinatorial algorithms, theory and practice. Englewood Cliffs, NJ: Prentice-Hall; 1977.

Resnikoff; H.L.; Dolby, J.L. The nature of affixing in written English. Mechanical Translation. 8(3,4): 84 and 89; 1965 June and October.

Resnikoff, H.L.; Dolby, J.L. The nature of affixing in written English, Part II. Mechanical Translation. 9(2): 23-33; 1966 June.

Rich, R.P.; Stone, A.G. Method for hyphenating at the end of a printed line. Communications of the ACM. 8(7): 444-445; 1965 July.

Riseman, Edward M.; Hanson, Allen R. A contextual postprocessing system for error correction using binary n-grams. IEEE Transactions on Computers. C-23(5): 480-493; 1974 May.

Salton, G.; Wong, A. Generation and search of clustered files. ACM Transactions on Data base Systems. 3(4): 321-346; 1978 December.

Shannon, E. E. A mathematical theory of communication. Bell System Technical Journal. 27: 379-423 and 623-656; 1948 July and October.

Sharpe, Richard B.; Fox, John B.; Hammond, Silas E. Computer-based editing of personalized corporate author, inventory, and patent assignee names for publication in CA author indexes. Brenner, Everett, comp. The Information age in perspective. Proceedings of the ASIS annual meeting; 1978 November 13-17; New York, N.Y. White Plains, N.Y.: Knowledge Industry Publications; 303-305; 1978.

Szanser, A.J. Bracketing technique in elastic matching. The Computer Journal. 16(2): 132-134; 1973.

Tagliacozzo, Renata; Kochen, Manfred; Rosenberg, Lawrence. Orthographic error patterns of author names in catalog searches. Journal of Library Automation 3(2): 93-101; 1970 June.

Ullman, J.R. A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words. The Computer Journal. 20(2): 141-147; 1977.

Van Nes, F.L. Analysis of keying errors. Ergonomics. 19(2): 165-174; 1976.

Wagner, Robert A.; Fischer, Michael J. The string-to-string correction problem. Journal of the ACM. 21(1): 168-178; 1974 January.

Willett, Peter. Document retrieval experiments using indexing vocabularies of varying size. II. Hashing, truncation, diagram and trigram encoding of index terms. Journal of Documentation. 35(4): 296-305; 1979 December.