AN ALGORITHM FOR VARIABLE-LENGTH PROPER-NAME COMPRESSION

James L. DOLBY: R & D Consultants Company, Los Altos, California

Viable on-line search systems require reasonable capabilities to automatically detect (and hopefully correct) variations between request format and stored format. An important requirement is the solution of the problem of matching proper names, not only because both input specifications and storage specifications are subject to error, but also because various transliteration schemes exist and can provide variant proper name forms in the same data base. This paper reviews several proper name matching schemes and provides an updated version of these schemes which tests out nicely on the proper name equivalence classes of a suburban telephone book. An appendix lists the corpus of names used for algorithm test.

A viable on-line search system cannot reasonably assume that each user will invariably provide the proper input information without error. Human beings not only make errors, but also expect their correspondents, be they human or mechanical, to be able to cope with these errors, at least at some reasonable error-rate level. Many of the difficulties in implementing computer systems in many areas of human activity stem from failure to recognize, and plan for, routine acceptance of errors in the systems. Indeed, computing did not become the widespread activity it is now until the so-called higher-level languages came into being. Although it is customary to think of higher-level languages as being "more English-like," the height of their level is better measured by the brevity with which various jobs can be expressed (for brevity tends to reduce errors) and the degree of sophistication of their automatic error detection and correction procedures.

The processing of catalog information for the purposes of exposing and retrieving information presents at least two major areas for research in automatic error detection and correction. At the first stage, the data bank must be created, updated and maintained. Methods for dealing with input errors at this level have been derived by a number of groups and it seems reasonable to assert that something in the order of 60% of the input errors can be detected automatically (1,2,3). With the possibility of human proof-

reading and error detection through actual use, it is reasonable to expect a mature data base to have a very low over-all error rate.

At the second stage, however, when a user approaches the data base through a terminal or other on-line device, the errors will be of a recurring nature. Each user will generate his own error set and, though experience will tend to minimize the error rate for a particular user, there will be an essentially irreducible minimum error rate even for an experienced user. If the system is to attract users other than professional interrogators, it must respond intelligently at this minimal error level.

This paper explores certain problems associated with making "noisy matches" in catalog searches. Because preliminary information indicates that the most likely source of input errors is in the keyboarding of proper names, the main emphasis of the paper is on the problem of algorithmically compressing proper names in such a way as to identify similar names (and likely misspellings) without over-identifying the list of possible authors.

EXISTING NAME-COMPRESSION ALGORITHMS

The problem of providing equivalence classes of proper names is hardly new. Library catalogs, telephone directories and other major data bases have made use of "see-also"-type references for many years. Some years ago Remington-Rand derived an alphanumeric name compression algorithm, SOUNDEX, that could be applied either by hand or by machine for such purposes (4). Perhaps the most widely used on-line retrieval system presently in existence, the airline reservation system (such as SABRE), makes use of such an algorithm (5). The closely related problem of compressing English words (either to establish noisy matches, to eliminate misspelled words, or simply to achieve data bank compression) has also received some attention (6, 7, 8). Implementation of such algorithms has been described (9, 10, 11, 12, 13).

Although English word structure differs from proper-name structure in some important respects (e.g., the existence of suffixes), three of the algorithms are constructed by giving varying degrees of attention to the following five areas of word structure:

1) The character in word initial position;
2) The character set: (A, E, I, O, U, Y, H, W);
3) Doubled characters (e.g., tt);
4) Transformation of consonants (i.e., all alphabetic characters other than those in 2 above) into equivalence classes;
5) Truncation of the residual character string.

The word-initial character receives varying attention. SOUNDEX places the initial consonant in the initial position of the compressed form and then transforms all other consonants into equivalence classes with numeric titles. SABRE maintains the word-initial character even if it is a vowel. In the Armour Research Foundation scheme (ARF), the word-initial character is also retained as is.

Both SOUNDEX and SABRE eliminate all characters in the set 2) above. The ARF scheme retains all characters in shorter words and deletes vowels only, to reduce the compressed form to four characters, deleting the "U" after "Q," the second vowel in a vowel string, and then all remaining vowels.

All three systems delete the second letter of a double-letter string. SABRE goes a step further and deletes the second letter of a double-letter string occurring after the vowels have been deleted. Thus, the second "R" of "BEARER" would be deleted.

SOUNDEX maps the eighteen consonants into six equivalence classes:

1) B, F, P, V
2) C, G, J, K, Q, S, X, Z
3) D, T
4) L
5) M, N
6) R

SABRE and ARF do not perform any transformations on these eighteen consonants.

Finally, all three systems truncate the remaining string of characters to four characters. For shorter forms, padding in the form of zeros (SOUNDEX), blanks (SABRE), or hyphens (ARF) is added so that all codes are precisely four characters long.

Variable-length coding schemes have been considered but generally rejected for implementation on major systems because of the attendant difficulties of programming and the fact that code compression is enhanced by fixed-length codes where no interword space is necessary. Although fixed-length schemes of length greater than four have been considered, no definitive data appears to be available as to the enhanced ability of compressed codes to discriminate by introduction of more characters. The SABRE system does add a fifth character but makes use of the person's first initial for added discrimination.

Tukey (14) has constructed a personal author code for his citation indexing and permuted title studies on an extensive corpus of the statistical literature. In this situation the author code is a semi-mnemonic code in a tag form to assist the user in identification rather than to be used as a basic entry point. However, Tukey does note that in his corpus a three-character code of the surname, plus two initials, is superior to a five-character surname code for purposes of unique identification.

MEASURING ALGORITHMIC PERFORMANCE

One of the main problems in constructing linguistic algorithms is to decide on appropriate measures of performance and to obtain data bases for implementing such measures. In this case it is clear that certain improvements in existing algorithms can be made — particularly by using more sophisticated transformation rules for the consonants — and that

the problems of implementing such changes are not so great in today's context as they were when the systems noted above were originally derived. Improvements in processing speeds and programming languages, however, do not remove the need for keeping "linguistic frills" to a minimum.

Ideally, it would be desirable to have a list of common errors in keyboarding names as a test basis for any proposed algorithms. Unfortunately, no such list of sufficient size appears to be available. Lacking this, one can speculate that certain formal properties of the predictability of language might be useful in deriving an algorithm. At the English word level, some effort has been made to exploit measures of entropy as developed by Shannon in this direction (6, 7). However, there is good reason to question whether entropy, at least when measured in the usual way, is strongly correlated with actually occurring errors (15).

As an alternative, one can study existing lists of personal-name equivalence classes to derive such algorithms and then test the algorithm against such classes, measuring both the degree of over-identification and the degree of under-identification. Clearly, such tests will carry more weight if they are conducted under economic forcing conditions where weaknesses in the test set will lead to real and measurable expense to the organization publishing the list. The SABRE system operates under strong economic forcing conditions in the sense that airline passengers frequently have a number of competitive alternatives available to them and lost reservations can cause sufficient inconvenience for them to consider these alternatives. However, the main application of the SABRE system is to rather small groups of persons (at least when compared to the number of personal authors in a typical library catalog), so that errors of over-identification are essentially trivial in cost to the airlines.

A readily available source of "see-also"-type equivalence classes of proper names is given in the telephone directory system. Here, the economic forcing system is not so strong as in the airline situation, but it is measurable in that failure to provide an adequate list will lead to increased user dependence on the Information Operator, with consequent increased cost to the telephone company. As a test of the feasibility of using such a set of equivalence classes, the 451 classes found in the Palo Alto-Los Altos (California) telephone directory were copied out by hand and used in deriving and testing the algorithm given in the next section and the SOUNDEX algorithm.

There remains the question of deciding what is to constitute proper agreement between any algorithm and the set of equivalence classes chosen as a data base. At the grossest level it seems reasonable to argue that over-identification is less serious than under-identification. False drops only tend to clog the line. Lost reference points, on the other hand, lead to lost information. Investigation of other applications of linguistic algorithms, such as algorithms to hyphenate words, identify semantically similar words through cutting off of suffixes, and so forth, indicates that it is usually

possible to reduce crucial error (in this case under-identification) to something under 5%, while preserving something in the order of 80% of the original distinctions (or efficiency) of the system. Efforts to improve materially on the "five-and-eighty" rule generally lead to solutions involving larger context and/or extensive exception dictionaries. In this study efforts are directed at achieving a "five-and-eighty" solution.

A VARIABLE-LENGTH NAME-COMPRESSION SCHEME

In light of the fact that no definitive information is available on the problems of truncating errors in name-compression algorithms, it is convenient to break the problem into two pieces. First is derivation of a variable-length algorithm of the required accuracy and efficiency and then determination of the errors induced by truncation.

A studying of the set of equivalence classes given in the Palo Alto-Los Altos telephone directory made fairly clear that with minor modifications of the basic five steps used in the other algorithms noted above, it would not be too difficult to provide a reasonably accurate match without requiring too much over-identification. The main modifications made consisted of maintaining the position of the first vowel and using local context to make transformations on the consonants. The algorithm is given below. (The rules given must be applied in the order given both with respect to the rules themselves and to the order of the lists within the rules, as the precedence relations are important to the performance of the algorithm.)

A Spelling Equivalent Abbreviation Algorithm For Personal Names

1) Transform: "McG" to "Mk", "Mag" to "Mk", "Mac" to "Mk", "Mc" to "Mk".
2) Working from the right, recursively delete the second letter from the following letter pairs: "dt", "ld", "nd", "nt", "rc", "rd", "rt", "sc", "sk", "st".
3) Transform: "x" to "ks", "ce" to "se", "ci" to "si", "cy" to "sy", "consonant-ch" to "consonant-sh"; all other occurrences of "c" to "k", "z" to "s", "wr" to "r", "dg" to "g", "qu" to "k", "t" to "d", "ph" to "f" (after the first letter).
4) Delete all consonants other than "l", "n", and "r" which precede the letter "k" (after the first letter).
5) Delete one letter from any doubled consonant.
6) Transform "pf#" to "p#", "#pf" to "#f", "vowel-gh#" to "vowel-f#", "consonant-gh" to "consonant-g#", and delete all other occurrences of "gh". ("#" is the word-beginning and word-ending marker.)
7) Replace the first vowel in the name by the symbol "*".
8) Delete all remaining vowels.
9) Delete all occurrences of "w" or "h" after the first letter in the word.

The vowels are taken to be (A, E, I, O, U, Y). The remaining literal characters are treated as consonants.

The algorithm splits 22 (4.9%) of the 451 equivalence classes given by the phone directory. On the other hand, the algorithm provides 349 distinct classes (not counting those classes that were broken off in error) or 77.4% of the 451 classes in the telephone directory data base. Thus has been achieved a reasonable approximation to the "five-and-eighty" performance found in other linguistic problem areas.

To give a proper appreciation of the nature of these underidentification errors, they are discussed below individually.

1) The name Bryer is put in the same equivalence class with a variety of spellings of the name Bear. The algorithm fails to make this identification.
2) Blagburn is not equated to Blackburn.
3) The name Davison is equated to Davidson in its various forms. The algorithm fails to make this identification and this appears to be one of a modest class of difficulties that occur prior to the -son, -sen names.
4) The class of names Dickinson, Dickerson, Dickison, and Dickenson are all equated by the directory but kept separate, except for the two forms of Dickinson, by the algorithm.
5) The name Holm is not equated with the name Home.
6) The name Holmes is not equated with the name Homes.
7) The algorithm fails to equate Jaeger with two forms of Yaeger.
8) The algorithm fails to equate Lamb with Lamn.
9) The algorithm incorrectly assumes that the final "gh" of Leigh should be treated as an "f." Treating final "gh" either as a null sound or an "f" leads to about the same number of errors in either direction.
10) The algorithm fails on the pairing of Leicester and Lester. The difficulty is an intervening vowel.
11) The algorithm fails to equate the various forms of Lindsay with the forms of Lindsley.
12) The algorithm fails to equate the various forms of McLaughlin with McLachlan.
13) The algorithm fails to equate McCullogh with McCullah. This is again the final "gh" problem.
14) The algorithm fails to equate McCue with McHugh (again the final "gh" problem).
15) The algorithm fails to equate Moretton with Morton. This is an intervening vowel problem.
16) The algorithm fails to equate Rauch with Roush.
17) The algorithm fails to equate Robinson with Robison (another -son type problem).
18) The algorithm incorrectly assumes that the interior "ph" of Shepherd is an "f."
19) The algorithm fails to equate Speer with Speier.

20) The algorithm fails to equate Stevens with Stephens.
21) The algorithm fails to equate Stevenson with Stephenson.
22) The algorithm fails to equate the various forms of the word Thompson (an -son problem.)

In several of the errors noted above it may be questioned whether the telephone directory is following its own procedures with complete rigor. Setting these aside, the primary errors occur with the final "gh," the words ending in "son," and the words with the extraneous interior vowels. Each of these problems can be resolved to any desired degree of accuracy, but only at the expense of noticeable increases in the degree of complexity of the algorithm.

THE TRUNCATION PROBLEM

Simple truncation does not introduce errors of under-identification; it can only lead to further over-identification. Examination of the results of applying the algorithm to the telephone directory data base shows that no new over-identification is introduced if the compressed codes are all reduced to the leftmost seven characters. Further truncation leads to the following results:

Code Length	*Cumulative Over-Identification Losses*
7	0
6	1
5	6
4	45

Thus there is a strong argument for maintaining at least five characters in the compressed code.

However, there is no real need for restriction to simple truncation. Following the procedures used in the ARF system, further truncation can be obtained by selectively removing some of the remaining characters. The natural candidate for such removal is the vowel marker. If the vowel marker is removed from all the five character codes, only six more over-identification errors are introduced. Removal of the vowel markers from all of the codes would have introduced 17 more errors of over-identification. The utility of the vowel marker is in the short codes. This in turn suggests that introduction of a second vowel marker in the very short codes may have some utility, and this is indeed the case. If the conception of vowel marker is generalized as marking the position of a vowel-string (i.e., a string of consecutive vowels), where for these purposes a vowel is any of the characters (A, E, I, O, U, Y, H, W), and these markers are maintained as "padding" in the very short words, 18 errors of over-identification are eliminated at the cost of two new errors of under-identification. In this way the following modification to the variable length algorithm is derived:

1) Mark the position of each of the first two vowel strings with an "*," if there is more than one vowel.

2) Truncate to six characters.
3) If the six-character code has two vowel markers, remove the right-hand vowel marker. Otherwise, truncate the sixth character.
4) If the resulting five-character code has a vowel marker, remove it. Otherwise remove the fifth character.
5) For all codes having less than four characters in the variable-length form, pad to four characters by adding blanks to the right.

Measured against the telephone directory data base, this fixed-length compression code provides 361 distinct classes (not counting improper class splits as separate classes) or 80% of the 451 given classes. Twenty-four (5.3 %) of the classes are improperly split. By way of comparison, the SOUNDEX system improperly splits 135 classes (30%) and provides only 287 distinct classes (not counting improperly split classes), or 63.8% of the telephone directory data base.

ACKNOWLEDGMENTS

This research was carried out for the Institute of Library Research, University of California, under the sponsorship of the Office of Education, Research Grant No. OEG-1-7-071083-5068.

The author would like to thank Ralph M. Shoffner and Kelley L. Cartwright for suggesting the problem and for a number of useful comments on existing systems. Allan J. Humphrey was kind enough to program the variable-length version of the algorithm for test purposes.

APPENDIX: CORPUS OF NAMES USED FOR ALGORITHM TEST

A list of personal-name equivalence classes from the Palo Alto-Los Altos Telephone Directory is arranged according to the variable-length compression code (with the vowel marked "*" treated as an "A" for ordering).

Names whose compressed codes do not match the one given in the first column (and hence represent weaknesses in the algorithm and/or the directory groupings) are given in italics.

A small number of directory entries that do not bear on the immediate problem have been deleted from the list: Bell's *see also* Bells; Co-op *see also* Co-operative; St. *see also* Saint; etc.

*BL	Abel, Abele, Abell, Able
*BRMS	Abrahams, Abrams
*BRMSN	Abrahamson, Abramson
*D	Eddy, Eddie
*DMNS	Edmonds, Edmunds
*DMNSN	Edmondson, Edmundson
*DMS	Adams, Addems
*GN	Eagen, Egan, Eggen
*GR	*Jaeger*, Yaeger, Yeager
*KN	Aiken, Aikin, Aitken
*KNS	Adkins, Akins

*KR	Acker, Aker
*KR	Eckard, Eckardt, Eckart, Eckert, Eckhardt
*KS	Oakes, Oaks, Ochs
*LBRD	Albright, Allbright
*LD	Elliot, Elliott
*LN	Allan, Allen, Allyn
*LSN	Ohlsen, Olesen, Olsen, Olson, Olsson
*LVR	Oliveira, Olivera, Olivero
*MS	Ames, Eames
*NGL	Engel, Engle, Ingle
*NL	O'Neal, O'Neil, O'Neill
*NRS	Andrews, Andrus
*NRSN	Andersen, Anderson, Andreasen
*NS	Ennis, Enos
*RKSN	Enrichsen, Erickson, Ericson, Ericsson, Eriksen
*RL	Earley, Early
*RN	Erwin, Irwin
*RNS	Aarons, Ahrends, Ahrens, Arens, Arentz, Arons
*RS	Ayers, Ayres
*RVN	Ervin, Ervine, Irvin, Irvine
*RVNG	Erving, Irving
*SBRN	Osborn, Osborne, Osbourne, Osburn
B*D	Beatie, Beattie, Beatty, Beaty, Beedie
B*DS	Betts, Betz
B*KMN	Bachman, Bachmann, Backman
B*L	Bailey, Baillie, Bailly, Baily, Bayley
B*L	Beal, Beale, Beall, Biehl
B*L	Belew, Ballou, Bellew
B*L	Buhl, Buell
B*L	Belle, Bell
B*LN	Bolton, Boulton
B*M	Baum, Bohm, Bohme
B*MN	Bauman, Bowman
B*N	Bain, Bane, Bayne
B*ND	Bennet, Bennett
B*R	Baer, Bahr, Baier, Bair, Bare, Bear, Beare, Behr, Beier, Bier, *Bryer*
B*R	Barry, Beare, Beery, Berry
B*R	Bauer, Baur, Bower
B*R	Bird, Burd, Byrd
B*RBR	Barbour, Barber
B*RG	Berg, Bergh, Burge
B*RGR	Berger, Burger
B*RK	Boerke, Birk, Bourke, Burk, Burke
B*RN	Burn, Byrne

B*RNR	Bernard, Bernhard, Bernhardt, Bernhart
B*RNS	Berns, Birns, Burns, Byrns, Byrnes
B*RNSN	Bernstein, Bornstein
B*RS	Bertsch, Birch, Burch
BL*KBRN	Blackburn, *Blagburn*
BL*M	Blom, Bloom, Bluhm, Blum, Blume
BR*D	Brode, Brodie, Brody
BR*N	Braun, Brown, Browne
BR*N	Brand, Brandt, Brant
D*DS	Diezt, Ditz
D*F	Duffie, Duffy
D*GN	Dougan, Dugan, Duggan
D*K	Dickey, Dicke
D*KNSN	Dickenson, *Dickerson*, Dickinson, *Dickison*
D*KSN	Dickson, Dixon, Dixson
D*L	Dailey, Daily, Daley, Daly
D*L	Dahl, Dahle, Dall, Doll
D*L	Deahl, Deal, Diehl
D*MN	Diamond, Dimond, Dymond
D*N	Dean, Deane, Deen
D*N	Denney, Denny
D*N	Donahoo, Donahue, Donoho, Donohoe, Donohoo, Donohue, Dunnahoo
D*N	Downey, Downie
D*N	Dunn, Dunne
D*NL	Donley, Donnelley, Donnelly
D*R	Daugherty, Doherty, Dougherty
D*R	Dyar, Dyer
D*RM	Derham, Durham
D*VDSN	Davidsen, Davidson, *Davison*
D*VS	Davies, Davis
DR*SL	Driscoll, Driskell
F*	Fay, Fahay, Fahey
F*FR	Fifer, Pfeffer, Pfeiffer
F*GN	Fagon, Feigan, Fegan
F*L	Feil, Pfeil
F*L	Feld, Feldt, Felt
F*LKNR	Faulkner, Falconer
F*LPS	Philips, Phillips
F*NGN	Finnegan, Finnigan
F*NL	Finlay, Finley
F*RL	Farrell, Ferrell
F*RR	Ferrara, Ferreira, Ferriera
F*RR	Foerster, Forester, Forrester, Forster
F*RS	Forrest, Forest

F*RS	Faris, Farriss, Ferris, Ferriss
F*RS	First, Fuerst, Furst
F*SR	Fischer, Fisher
FL*N	Flinn, Flynn
FL*NGN	Flanagan, Flanigan, Flannigan
FR*	Frei, Frey, Fry, Frye
FR*DMN	Freedman, Friedman
FR*DRKSN	Frederickson, Frederiksen, Fredickson, Fredriksson
FR*K	Franck, Frank
FR*NS	France, Frantz, Franz
FR*NS	Frances, Francis
FR*S	Freeze, Freese, Fries
FR*SR	Fraser, Frasier, Frazer, Frazier
G*D	Good, Goode
G*DS	Getz, Goetz, Goetze
G*F	Goff, Gough
G*L	Gold, Goold, Gould
G*LMR	Gilmer, Gilmore, Gilmour
G*LR	Gallagher, Gallaher, Galleher
G*MS	Gomes, Gomez
G*NR	Guenther, Gunther
G*NSLS	Gonzales, Gonzalez
G*NSLVS	Consalves, Gonzalves
G*RD	Garratt, Garrett
G*RD	Garrity, Geraghty, Geraty, Gerrity
G*RN	Gorden, Gordohn, Gordon
G*RNR	Gardiner, Gardner, Gartner
G*RR	Garrard, Gerard, Gerrard, Girard
G*S	Gauss, Goss
GR*	Gray, Grey
GR*FD	Griffeth, Griffith
GR*N	Green, Greene
GR*S	Gros, Grose, Gross
H*D	Hyde, Heidt
H*F	Hoff, Hough, Huff
H*FMN	Hoffman, Hoffmann, Hofman, Hofmann, Huffman
H*G	Hoag, Hoge, Hogue
H*GN	Hagan, Hagen
H*K	Hauch, Hauck, Hauk, Hauke
H*KSN	Hutcheson, Hutchison
H*L	Holley, Holly
H*L	Holl, Hall
H*L	Halley, Haley
H*L	Haile, Hale
H*LD	Holiday, Halliday, Holladay, Holliday

H°LG	Helwig, Hellwig
H°LM	Holm, *Home*
H°LMS	Holmes, *Homes*
H°LN	Highland, Hyland
H°M	Ham, Hamm
H°MR	Hammar, Hammer
H°N	Hanna, Hannah
H°N	Hahn, Hahne, Hann, Haun
H°NN	Hanan, Hannan, Hannon
H°NRKS	Hendricks, Hendrix, Henriques
H°NRKSN	Hendrickson, Henriksen, Henrikson
H°NS	Heintz, Heinz, Heinze, Hindes, Hinds, Hines, Hinze
H°NS	Haines, Haynes
H°NSN	Henson, Hansen, Hanson, Hanssen, Hansson, Hanszen
H°R	Herd, Heard, Hird, Hurd
H°R	Hart, Hardt, Harte, Heart
H°R	Hare, Hair
H°R	Hardey, Hardie, Hardy
H°RMN	Hartman, Hardmen, Hardmon, Hartmann
H°RMN	Herman, Hermann, Herrmann
H°RMN	Harman, Harmon
H°RN	Heron, Herrin, Herron
H°RN	Hardin, Harden
H°RN	Horn, Horne
H°RNGDN	Herrington, Harrington
H°S	Haas, Haase, Hasse
H°S	Howes, House, Howse
H°S	Hays, Hayes
H°SN	Houston, Huston
H°VR	Hoover, Hover
J°	Jew, Jue
J°FR	Jeffery, Jeffrey
J°FRS	Jefferies, Jefferis, Jefferys, Jeffreys
J°KB	Jacobi, Jacoby
J°KBSN	Jacobsen, Jacobson, Jackobsen
J°KS	Jacques, Jacks, Jaques
J°L	Jewell, Juhl
J°MS	Jaimes, James
J°MSN	Jameson, Jamieson, Jamison
J°NSN	Jahnsen, Jansen, Jansohn, Janssen, Jansson, Janzen, Jensen, Jenson
J°S	Joice, Joyce
K°	Kay, Kaye
K°F	Coffee, Coffey
K°FMN	Coffman, Kauffman, Kaufman, Kaufmann

K*K	Cook, Cooke, Koch, Koche
K*L	Cole, Kohl, Koll
K*L	Kelley, Kelly
K*LMN	Coleman, Colman
K*LR	Koehler, Koeller, Kohler, Koller
K*MBRLN	Chamberlain, Chamberlin
K*MBS	Combs, Coombes, Coombs
K*MP	Camp, Kampe, Kampf
K*MPS	Campos, Campus
K*N	Cahn, Conn, Kahn
K*N	Cahen, Cain, Caine, Cane, Kain, Kane
K*N	Chin, Chinn
K*N	Chaney, Cheney
K*N	Coen, Cohan, Cohen, Cohn, Cone, Koehn, Kohn
K*N	Coon, Kuhn, Kuhne
K*N	Kenney, Kenny, Kinney
K*NL	Conley, Conly, Connelly, Connolly
K*NR	Conner, Connor
K*NS	Coons, Koontz, Kuhns, Kuns, Kuntz, Kunz
K*P	Coop, Co-op, Coope, Coupe, Koop
K*PL	Chapel, Chapell, Chappel, Chappell, Chappelle, Chapple
K*R	Carrie, Carey, Cary
K*R	Corey, Cory
K*R	Carr, Kar, Karr
K*R	Kurtz, Kurz
K*R	Kehr, Ker, Kerr
K*RD	Cartwright, Cortright
K*RLN	Carleton, Carlton
K*RN	Carney, Cerney, Kearney
K*RSNR	Kirschner, Kirchner
K*S	Chace, Chase
K*S	Cass, Kass
K*S	Kees, Keyes, Keys
K*SL	Cassel, Cassell, Castle
K*SLR	Kesler, Kessler, Kestler
K*SR	Kaiser, Kayser, Keizer, Keyser, Kieser, Kiser, Kizer
KL*N	Cline, Klein, Kleine, Kline
KL*RK	Clark, Clarke
KL*SN	Claussen, Clausen, Clawson, Closson
KR*	Crow, Crowe
KR*GR	Krieger, Kroeger, Krueger, Kruger
KR*MR	Creamer, Cramer, Kraemer, Kramer, Kremer
KR*N	Craine, Crane
KR*S	Christie, Christy, Kristee
KR*S	Crouss, Kraus, Krausch, Krause, Krouse

KR°S	Cross, Krost
KR°S	Crews, Cruz, Kruse
KR°SNSN	Christensen, Christiansen, Christianson
L°	Loe, Loewe, Low, Lowe
L°	Lea, Lee, *Leigh*
L°D	Lloyd, Loyd
L°DL	Litle, Littell, Little, Lytle
L°DRMN	Ledterman, Letterman
L°K	Leach, Leech, Leitch
L°KS	Lucas, Lukas
L°LN	Laughlin, Loughlin
L°LR	Lawler, Lawlor
L°MB	Lamb, *Lamm*
L°MN	Lemen, Lemmon, Lemon
L°MN	Layman, Lehman, Lehmann
L°N	Lind, Lynd, Lynde
L°N	Lion, Lyon
L°N	Lin, Linn, Lynn, Lynne
L°N	Lain, Laine, Laing, Lane, Layne
L°NG	Lang, Lange
L°NN	London, Lundin
L°NS	Lindsay, Lindsey, *Lindsley, Linsley*
L°R	Lawry, Lowery, Lowrey, Lowry
L°RNS	Lawrence, Lowrance
L°RNS	Laurence, Lawrance, Lawrence, Lorence, Lorenz
L°RSN	Larsen, Larson
L°S	Lewis, Louis, Luis, Luiz
L°S	Lacey, Lacy
L°SR	*Leicester,* Lester
L°V	Levey, Levi, Levy
L°VD	Leavett, Leavitt, Levit
L°VL	Lavell, Lavelle, Leavelle, Loveall, Lovell
L°VN	Lavin, Levin, Levine
M°D	Mead, Meade
M°DN	*Moretton,* Morton
M°DS	Mathews, Matthews
M°DSN	Madison, Madsen, Matson, Matteson, Mattison, Mattson
M°KL	Michael, Michel
M°KM	Meacham, Mechem
M°KS	Marques, Marquez, Marquis, Marquiss
M°KS	Marcks, Marks, Marx
M°LN	Maloney, Moloney, Molony
M°LN	Mullan, Mullen, Mullin
M°LR	Mallery, Mallory
M°LR	Moeller, Moller, Mueller, Muller

M°LR	Millar, Miller
M°LS	Miles, Myles
M°N	Mahan, Mann
M°NR	Miner, Minor
M°NR	Monroe, Munro
M°NSN	Monson, Munson
M°R	Murray, Murrey
M°R	Maher, Maier, Mayer
M°R	Mohr, Moor, Moore
M°R	Meyers, Myers
M°R	Meier, Meyer, Mieir, Myhre
M°RF	Murphey, Murphy
M°RL	Merrell, Merrill
M°RN	Marten, Martin, Martine, Martyn
M°RS	Meyers, Myers
M°RS	Maurice, Morris, Morse
MK°	McCoy, McCaughey
MK°	Magee, McGee, McGehee, McGhie
MK°	Mackey, MacKay, Mackie, McKay
MK°	McCue, *McHugh*
MK°L	Magill, McGill
MK°LF	McCollough, *McCullah,* McCullough
MK°LM	McCallum, McCollum, McColm
MK°N	McKenney, McKinney
MK°NR	MacIntyre, McEntire, McIntire, McIntyre
MK°NS	MacKenzie, McKenzie
MK°NS	Maginnis, McGinnis, McGuinness, McInnes, McInnis
MK°R	Maguire, McGuire
MK°R	McCarthy, McCarty
MKD°NL	MacDonald, McDonald, McDonnell
MKF°RLN	MacFarland, MacFarlane, McFarland, McFarlane
MKF°RSN	MacPherson, McPherson
MKL°D	MacLeod, McCloud, McLeod
MKL°KLN	MacLachlan, Maclachlin, McLachlan, *McLaughlin, McLoughlin*
MKL°LN	McClellan, McClelland, McLellan
MKL°N	McClain, McClaine, McLain, McLane
MKL°N	MacLean, McClean, McLean
MKL°S	McCloskey, McClosky, McCluskey
MKM°LN	MacMillan, McMillan, McMillin
MKN°L	MacNeal, McNeal, McNeil, McNeill
MKR°D	Magrath, McGrath
N°KL	Nichol, Nicholl, Nickel, Nickle, Nicol, Nicoll
N°KLS	Nicholls, Nichols, Nickels, Nickles, Nicols
N°KLS	Nicholas, Nicolas

N*KLSN	Nicholsen, Nicholson, Nicolaisen, Nicolson
N*KSN	Nickson, Nixon
N*L	Neal, Neale, Neall, Neel, Neil, Neill
N*LSN	Neilsen, Neilson, Nelsen, Nelson, Nielsen, Nielson, Nilson, Nilssen, Nilsson
N*MN	Neumann, Newman
N*RS	Norris, Nourse
N*SBD	Nesbit, Nesbitt, Nisbet
P*D	Pettee, Petty
P*DRSN	Peterson, Pederson, Pedersen, Petersen, Petterson
P*G	Page, Paige
P*LK	Polak, Pollack, Pollak, Pollock
P*LSN	Polson, Paulsen, Paulson, Poulsen, Poulsson
P*N	Paine, Payn, Payne
P*R	Parry, Perry
P*R	Parr, Paar
P*RK	Park, Parke
P*RKS	Parks, Parkes
P*RS	Pierce, Pearce, Peirce, Piers
P*RS	Parish, Parrish
P*RS	Paris, Parris
P*RSN	Pierson, Pearson, Pehrson, Peirson
PR*KR	Prichard, Pritchard
PR*NS	Prince, Prinz
PR*R	Prior, Pryor
R*	Roe, Rowe
R*	Rae, Ray, Raye, Rea, Rey, Wray
R*BNSN	Robinson, *Robison*
R*D	Rothe, Roth
R*D	Rudd, Rood, Rude
R*D	Reed, Read, Reade, Reid
R*DR	Rider, Ryder
R*DS	Rhoades, Rhoads, Rhodes
R*GN	Regan, Ragon, Reagan
R*GR	Rodgers, Rogers
R*K	Richey, Ritchey, Ritchie
R*K	Reich, Reiche
R*KR	Reichardt, Richert, Rickard
R*L	Reilley, Reilly, Reilli, Riley
R*MNGTN	Remington, Rimington
R*MR	Reamer, Reimer, Riemer, Rimmer
R*MS	Ramsay, Ramsey
R*N	Rhein, Rhine, Ryan
R*NR	Reinhard, Reinhardt, Reinhart, Rhinehart, Rinehart
R*S	Reas, Reece, Rees, Reese, Reis, Reiss, Ries

R°S	*Rauch,* Rausch, *Roach, Roche,* Roush
R°S	Rush, Rusch
R°S	Russ, Rus
R°VS	Reaves, Reeves
S°BR	Seibert, Siebert
S°FL	Schofield, Scofield
S°FN	Stefan, Steffan, Steffen, Stephan, Stephen
S°FNS	Steffens, Stephens, *Stevens*
S°FNSN	Steffensen, Steffenson, Stephenson, *Stevenson*
S°FR	Schaefer, Schaeffer, Schafer, Schaffer, Schafer, Shaffer, Sheaffer
S°FR	Stauffer, Stouffer
S°GL	Siegal, Sigal
S°GLR	Sigler, Ziegler
S°K	Schuck, Shuck
S°KS	Sachs, Sacks, Saks, Sax, Saxe
S°L	Seeley, Seely, Seley
S°L	Schell, Shell
S°LR	Schuler, Schuller
S°LS	Schultz, Schultze, Schulz, Schulze, Shults, Shultz
S°LV	Silva, Sylva
S°LVR	Silveira, Silvera, Silveria
S°MKR	Schomaker, Schumacher, Schumaker, Shoemaker, Shumaker
S°MN	Simon, Symon
S°MN	Seaman, Seemann, Semon
S°MRS	Somers, Sommars, Sommers, Summers
S°MS	Simms, Sims
S°N	Stein, Stine
S°N	Sweeney, Sweeny, Sweney
S°NR	Senter, Center
S°NRS	Sanders, Saunders
S°PR	Shepard, *Shephard, Shepheard, Shepherd,* Sheppard
S°R	Stahr, Star, Starr
S°R	Stewart, Stuart
S°R	Storey, Story
S°R	Saier, Sayre
S°R	Schwartz, Schwarz, Schwarze, Swartz
S°RL	Schirle, Shirley
S°RLNG	Sterling, Stirling
S°RMN	Scheuermann, Schurman, Sherman
S°RN	Stearn, Stern
S°RR	Scherer, Shearer, Sharer, Sherer, Sheerer
S°S	Sousa, Souza
SM°D	Smith, Smyth, Smythe

SM°D	Schmid, Schmidt, Schmit, Schmitt, Smit
SN°DR	Schneider, Schnieder, Snaider, Snider, Snyder
SN°L	Schnell, Snell
SP°LNG	Spalding, Spaulding
SP°R	Spear, Speer, *Speirer*
SP°R	Spears, Speers
SR°DR	Schroder, Schroeder, Schroeter
SR°DR	Schrader, Shrader
T°D	Tait, Tate
T°MSN	Thomason, *Thompson,* Thomsen, Thomson, Tomson
T°RL	Terrel, Terrell, Terrill
TR°S	Tracey, Tracy
V°L	Vail, Vaile, Vale
V°L	Valley, Valle
V°R	Vieira, Vierra
W°D	White, Wight
W°DKR	Whitacre, Whitaker, Whiteaker, Whittaker
W°DL	Whiteley, Whitley
W°DMN	Whitman, Wittman
W°DR	Woodard, Woodward
W°DRS	Waters, Watters
W°GNR	Wagener, Waggener, Wagoner, Wagner, Wegner, Waggoner
W°L	Willey, Willi
W°L	Wiley, Wylie
W°L	Wahl, Wall
W°LBR	Wilber, Wilbur
W°LF	Wolf, Wolfe, Wolff, Woolf, Woulfe, Wulf, Wulff
W°LKNS	Wilkens, Wilkins
W°LKS	Wilkes, Wilks
W°LN	Whalen, Whelan
W°LR	Walter, Walther, Wolter
W°LRS	Walters, Walthers, Wolters
W°LS	Wallace, Wallis
W°LS	Welch, Welsh
W°LS	Welles, Wells
W°LSN	Willson, Wilson
W°N	Winn, Wynn, Wynne
W°R	Worth, Wirth
W°R	Ware, Wear, Weir, Wier
W°RL	Wehrle, Wehrlie, Werle, Worley
W°RNR	Warner, Werner
W°S	Weis, Weiss, Wiese, Wise, Wyss
W°SMN	Weismann, Weissman, Weseman, Wiseman, Wismonn, Wissman

REFERENCES

1. Cox, N.S.M.; Dolby, J. L.: "Structured Linguistic Data and the Automatic Detection of Errors." In *Advances in Computer Typesetting* (London: Institute of Printing, 1966), pp. 122-125.
2. Cox, N.S.M.; Dews, J. D.; Dolby, J. L.,: *The Computer and the Library* (Hamden, Conn.: Archon Press, 1967).
3. Dolby, J. L.; Forsyth, V. J.; Resnikoff, H. L.: *Computerized Library Catalogs: Their Growth, Cost and Utility* (Cambridge, Massachusetts: The M.I.T. Press, 1969).
4. Becker, Joseph; Hayes, Robert M.: *Information Storage and Retrieval* (New York: Wiley, 1963), p. 143.
5. Davidson, Leon: "Retrieval of Misspelled Names in Airlines Passenger Record System," *Communications of the ACM,* 5 (1962), 169-171.
6. Blair, C. R.: "A Program for Correcting Spelling Errors," *Information & Control,* 3 (1960), 60-67.
7. Schwartz, E. S.: *An Adaptive Information Transmission System Employing Minimum Redundancy Word Codes* (Armour Research Foundation Report, April 1962). (AD 274-135).
8. Bourne, C. P.; Ford, D.: "A Study of Methods for Systematically Abbreviating English Words and Names," *Journal of the ACM,* 8 (1961), 538-552.
9. Kessler, M. M., "The "On-Line" Technical Information System at M.I.T.", in *1967 IEEE International Convention Record.* (New York: Institute of Electrical and Electronic Engineers, 1967), pp. 40-43.
10. Kilgour, F. G.: "Retrieval of Single Entries from a Computerized Library Catalog File," *American Society for Information Science, Proceedings,* 5 (1968), 133-136.
11. Nugent, W. R.: "Compression Word Coding Techniques for Information Retrieval," *Journal of Library Automation,* 1 (December 1968), 250-260.
12. Rothrock, H. I.: *Computer-Assisted Directory Search; A Dissertation in Electrical Engineering,* (Philadelphia: University of Pennsylvania, 1968).
13. Ruecking, F. H.: "Bibliographic Retrieval from Bibliographic Input; The Hypothesis and construction of a Test," *Journal of Library Automation,* 1 (December 1968), 227-238.
14. Tukey, J. W.: *A Tagging System for Journal Articles and Other Citable Items: A Status Report* (Princeton, N.J.: Statistical Techniques Research Group, Princeton University, 1963).
15. Resnikoff, H. L.; Dolby, J. L.: *A Proposal to Construct a Linguistic and Statistical Programming System,* (Los Altos, Cal.: R & D Consultants Company, 1967).