COMPRESSION WORD CODING TECHNIQUES FOR INFORMATION RETRIEVAL

William R. NUGENT: Vice President, Inforonics, Inc., Cambridge, Massachusetts

A description and comparison is presented of four compression techniques for word coding having application to information retrieval. The emphasis is on codes useful in creating directories to large data files. It is further shown how differing application objectives lead to differing measures of optimality for codes, though compression may be a common quality.

INTRODUCTION

Cryptographic studies have documented much useful language data having application to retrieval coding. Because unclassified cryptographic studies are few, Fletcher Pratt's 1939 work (1) remains the classic in its field. Gaines (2) has the virtue of being in print, and the more recent cryptographic history of Kahn (3), while comprehensive, lacks the statistical data that made the earlier works valuable. The word coding problem for language processing, as opposed to cryptography, has been extensively studied by Nugent and Vegh (4). Information theorists have contributed the greatest volume of literature on coding and have added to its mathematical basis, largely from the standpoint of communications and error avoidance.

A brief discussion of compression codes and their objectives is here presented, and then a description of a project encompassing four compression codes having application to retrieval directories. Two of the codings are newly devised. One is Transition Distance Coding, a randomizing code that results in short codes of high resolving power.

The second is Alphacheck. It combines high readability with good resolution, and permits simple truncation to be used by means of applying a randomized check character that acts as a surrogate of the omitted portion. It appears to have the greatest potential, in directory applications, of the codes considered here.

Recursive Decomposition is a selected letter code devised by the author several years ago (4). It has been tested and has the advantages of simple derivation and high resolution.

Soundex(5) is the only compression code that has achieved wide usage. It was devised at Remington Rand for name matching under conditions of uncertain spelling.

OBJECTIVES OF COMPRESSION CODING

It is desired to transform sets of variable length words into fixed length codes that will maximally preserve word to word discrimination. In the final directories to be used, the codes for several elements will be accessible to enable the matching of several factors before a file record is selected. The separate codes for differing factors need not be the same length, though each type of code will be of uniform length; nor need the codes for differing factors be derived by the same process.

What we loosely call codes, must be formally designated ciphers. That is, they must be derivable from the data words themselves, and not require "code books" to determine equivalences. This is so because the file directories must be derivable from file items, entries in directory form must be derivable from an input query, and these two directory items must match when a record is to be extracted. The ciphers need not be decipherable for the application under consideration, and in general are not.

Fixed length codes which provide the rough equivalent and simplicity of a margin entry in a paper directory, are generally desirable for machine directories.

The functions of the codes will determine their form, and a code or file key designed to meet one objective will generally not be satisfactory for any other objective. The following typical objectives serve as four examples:

(1) Create a file key for extraction of records in approximate file order, as is required for the common Sorting and Printout Problem. A typical code construction rule is to take the first six letters.

JOHNSEN ―► JOHNSE
JOHNSON ―► JOHNSO
JOHNSTON ―► JOHNST
JOHNSTONE ―► JOHNST

(2) Create a file key for extraction of records under conditions of uncertainty of spelling (airline reservation problem). A typical code construction rule is Vowel Elimination or Soundex. A typical matching rule is best match.

Vowel Elimination	*Soundex*
JOHNSEN —► JHNSN	J525 —► J52
JOHNSON —► JHNSN	J525 —► J52
JOHNSTON —► JHNSTN	J5235 —► J52
JOHNSTONE —► JHNSTN	J5235 —► J52

(3) Create a file key extraction of records from accurate input, with objective of maximum discrimination of similar entries (cataloging search problem). Typical code construction rules are Recursive Decomposition Coding or Transition Distance Coding.

Recursive Decomposition	*Transition Distance*
JOHNSEN —► JHNSEN	BFTZ
JOHNSON —► JHNSON	DNWU
JOHNSTON —► JHSTON	ZIKY
JOHNSTONE —► JHSONE	ECRC

For the file keys of primary concern accurate imput data is assumed and the objective is maximum discrimination. Desirably, a code would be as discriminating as Transition Distance Coding and be as readable as truncation coding. This can be achieved to some degree by combining the two codes into one, with an initial portion truncated and a final check character representing the remainder via a compressed Transition Distance Code: Alphacheck.

(4) Create a file key for human readability and high word to word discrimination. Possible code construction rules are Alphacheck, and simple truncation plus a terminal check character.

JOHNSEN —► JOHNSV
JOHNSON —► JOHNSX
JOHNSTON —► JOHNSD
JOHNSTONE —► JOHNS3

METHODS

The algorithms for creating the preceding codes are described in the following sections.

It is axiomatic that randomizing codes give the greatest possible discrimination for a given code space. The whole trick of creating a good compression code is to eliminate the natural redundancy of English orthography, and preserve discrimination in a smaller word size.

Letter-selection codes can only half accomplish this, due to the skewed distribution of letter usage. They can eliminate the higher-frequency components, but they cannot increase the use of the lower-frequency components.

Randomizing codes—often called "hash" codes, properly quasi-random codes—can equalize letter usage and hence make best use of the code space. Prime examples here are the variants of Gödel coding devised by Vegh(4) in which the principle of obtaining uniqueness via the products of unrepeated primes is exploited, as it is in the randomizing codes considered here. The problem in design of a randomizing code is that the results can be skewed rather than uniformly distributed due to the skewed nature of the letters and letter sequences that the codes operate on.

In Transition Distance Coding, the natural bias of letters and letter sequences is overcome by operating on a word parameter that is itself semi-random in nature. The following principle, not quite a theorem, applies: "Considering letters in their normal ordinal alphabetic position, and considering letter transitions to be unidirectional and cyclic, the distribution of transition distances in English words is essentially uniform."

In view of the fact that letter usage has an extremely skewed distribution, with a probability ratio in excess of 170 to one for the extremes, it is seen that the more uniform parameter of transition distances is a superior one for achieving randomized codes. The relative uniformity of transition distance needs further investigation, but one typical letter diagram sample from Gaines(2) with 9999 transitions (means number of occurrences of each distance = 385) yielded a mean deviation of 99 and a standard deviation of 123, and an extreme probability ratio of 3.3 to one for the different transition distances from 0 to 25. The distribution can be made more uniform by letter permutation. Permutation is used in the algorithm for Transition Distance Coding but not in Alphacheck.

Algorithm

The method of Transition Distance Coding is used to operate on a variable length word to achieve fixed length alphabetic or alphanumeric codes that exhibit quasi-random properties. The code is formed from the modulo product of primes associated with transition distances of permuted letters. The method is intended strictly for computer operation, as it is a simple program but an extremely tedious manual operation. There are five steps:

(1) Permute characters of natural language word. This breaks the diagram dependency that could make the transition distances less uniformly distributed. This step might be dispensed with if the resulting distributions prove satisfactory without it. The permutation process consists of taking the middle letter (or letter right of middle for words with an even number of letters), the first, the last, the second, the next-to-last, etc.

until all letters have been used. That is, for a letter sequence:
$a_1, a_2, \ldots a_i \ldots, a_n$
The following permutation is taken:

$$a_{\mathrm{Int}}\left(\frac{n}{2}+1\right), a_1, a_n, a_2, a_{n-1}, \ldots a_{(1+i)}, a_{(n-i)}, \ldots a_{\mathrm{Int}}\left(\frac{n}{2}+1\right) +_{4\ \mathrm{Rem}}\left(\frac{n}{2}\right)$$

where Int and Rem refer to the integer part and remainder, respectively. To illustrate a typical case:
JOHNSEN $\longrightarrow$ N J N O E H S

(2) Take transition distances of the characters. Assign letters a position value corresponding to their normal ordinal alphabetic positions excepting Z, which is equated to 0, (e.g., A=1, Y=25, Z=O), and take the transition distances between successive letters of the input sequence. Distance is measured unidirectionally in alphabetic order, and cyclically ("around the bend," Z to A.) The sequence AX has the transition distance N_X-N_A=24−1=23. Negative distances are converted to their positive cyclic distance by taking their 26 complement. That is, the sequence XB has the transition distance N_B− N_X=2−24=−22→ 26−22=4. To follow the 'JOHNSEN' example:
NJNOEHS $\longrightarrow$ (14,10,14,15,5,8,19) $\longrightarrow$ (22,4,1,16,3,11)
letter numbers distances

(3) Associate with each transition distance a corresponding prime number. Table 1 shows the primes corresponding to the transition distances, beginning with 5 so that the alphanumeric base (36) and all numbers are relatively prime. Following the example above:
(22,4,1,11,3,11) $\longrightarrow$ (89,13,5,61,11,41)
distances primes

(4) Multiply these primes, modulo the capacity of the computer. Integer multiplication in single precision is effected, disregarding overflow. For a computer with an 18-bit word length containing a 1-bit sign position, multiply modulo 2^{17}. That is, disregard product portions that equal or exceed 131,072. For a machine of this type, then, there will be generated a quasi-random number in the range of 0 to 131, 071. This is converted to alphanumeric form in the next step. Following the example:
89 x 13 x 5 x 61 = (352,885) Mod2^{17} = 90,741
90,741 x 11 = (998,151) Mod2^{17} = 80,647
80,647 x 41 = (3,306,527) Mod2^{17} = 39,727

(5) Convert to alphabetic or alphanumeric form. Now express the number derived above as an integer base 26 (alphabetic form) or base 36 (alphanumeric form) using a 4-digit code. In the case of alphabetic representation use the letters to represent the numbers of their ordinal position (A=1, B=2,

etc.), and use Z as zero. In alphanumeric form one would use the digits 0 to 9 to represent this range, and the letters A through Z would represent the range from 10 to 35. Using the 18-bit word length assumed, the alphabetic form is as good as the alphanumeric. The range of the random number extends to 131,071; the range of four-digit alphabetic representation extends to $(26^4-1) = 256{,}975$; the range of 4-digit alphanumeric representation extends to $(36^4-1) = 1{,}676{,}615$. Hence, the alphabetic representation is sufficient. Divide the random number successively by 26^3, 26^2, 26^1, and 26^0 to obtain the alphabetic form. For example:

$39{,}727/26^3 = 2 + \text{Rem } 4575 \longrightarrow \text{B}$
$4575/26^2 = 6 + \text{Rem } 520 \longrightarrow \text{F}$
$520/26^1 = 20 + \text{Rem } 0 \longrightarrow \text{T}$
$0/26^0 = 0 \longrightarrow \text{Z}$
JOHNSEN $\longrightarrow$ BFTZ

Alphacheck

Alphacheck is a means for creating a randomized alphanumeric check digit. When used with a selected letter compression code, it operates on the missing letters to generate a single character surrogate. It is used to add discrimination to a simple truncation code, in the hope of attaining a compression code that is both readable and resolving.

A process practically identical to that of Transition Distance Coding is used, except that at the final step the random number is taken modulo 36 and expressed as an alphanumeric character. The ten numeric digits represent themselves, and the letters A to Z represent the mod 36 numbers from 10 to 35, or their ordinal alphabetic value plus nine.

In this case, the difference between an alphabetic representation and an alphanumeric one is significant, since only one character is used, and the range of the Alphacheck character is much smaller than the range of the binary random number it is derived from.

The probability of no repetition of Alphacheck codes in a sample of size r, is a case of determining the probability of uniqueness for sampling with replacement from a population n, for which:

$$p = \frac{n!}{n^r\,(n-r)!}$$

where n is the range of the code, for alphanumeric Alphacheck, $n = 36$.

The median of the distribution of p, r_m gives the sample size for which the probability of uniqueness is 0.5. This is estimated by taking the logarithmic form of p, which yields a good approximation when n is large with respect to r.

$$\text{Ln } p \approx -\frac{r^2}{2n} \;;\; r_m \approx [2n \text{ Ln}(.5)]^{\frac{1}{2}} = 1.18\, n^{\frac{1}{2}} = 7.08$$

By comparison, r_m for n=26 is 6.05; for n=131.072 (Transition Distance Coding in 4 characters and modulo 2^{17}) r_m is 427.

One may conclude that the alphanumeric Alphacheck (36 symbol) has a 50% expectation of uniquely resolving seven otherwise identical five-letter truncations of source words. It offers a one-word advantage over the 26-symbol alphabetic Alphacheck.

Algorithm

It is not appropriate to use the identical randomizing method of T.D.C. (Transition Distance Coding), since this was designed to operate on full words, because it is desirable to operate on the omitted remainders of truncated words, which are often as short as two letters. When a two-letter remainder exists only one transition distance is involved, and hence only one prime number; and the individual primes are not uniformly dis-

Table 1. Letter Positions and Primes used in Transition Distance Coding and Alphacheck.

Letter	*Letter Position and Distance Value*	*Prime Number*
A	1	5
B	2	7
C	3	11
D	4	13
E	5	17
F	6	19
G	7	23
H	8	29
I	9	31
J	10	37
K	11	41
L	12	43
M	13	47
N	14	53
O	15	59
P	16	61
Q	17	67
R	18	71
S	19	73
T	20	79
U	21	83
V	22	89
W	23	97
X	24	101
Y	25	103
Z	0	107

tributed modulo 36. Hence, in the case where only one transition distance exists, the corresponding prime is multiplied by two additional primes corresponding to the letters involved (Table 1.) If only two distances are involved, associate another prime corresponding to the last letter. Since randomization is created largely by the multiplicative properties of the process, at least three factors are multiplied in all cases. Except for this difference in step three, the randomizing process is essentially identical to that of TDC. The steps are:

(1) If word is six letters or less take whole word; otherwise, take first five letters and compute an Alphacheck character for the sixth, based on the omitted letters.

(2) Take transition distances of the omitted letters (as in TDC).

(3) Associate with each transition distance a corresponding prime number (as in TDC). If only one transition distance exists, additionally associate prime numbers with the remaining letters. If only two transition distances exist, additionally associate a prime number with the last letter.

(4) Multiply these primes, modulo the capacity of the computer (as in TDC).

(5) Convert to alphanumeric form in 1 symbol, modulo 36, in which $0 \longrightarrow 1, \ldots, 9 \longrightarrow 9, 10 \longrightarrow A, 11 \longrightarrow B, \ldots, 35 \longrightarrow Z$.

The example of the JOHNS—names, shown in Table 2 illustrates the process.

Table 2. Example of Key Generation by Alphacheck.

Name	JOHNSEN	JOHNSON	JOHNSTON	JOHNSTONE
Truncated Portion	JOHNS	JOHNS	JOHNS	JOHNS
Remainder	EN	ON	TON	TONE
Letter #	5,14	15,14	20,15,14	20,16,14,5
Distance	9	25	21,25	21,25,17
Distance Primes	31	103	83,103	83,103,67
Letter Primes	17,53	59,53	53	–
Product	27,931	322,081	453,097	572,783
Mod 2^{17}	27,931	59,937	59,881	48,495
Mod 36	31	33	13	3
Alphacheck Character	V	X	D	3
Resulting Code	JOHNSV	JOHNSX	JOHNSD	JOHNS3

Recursive Decomposition Coding

This method uses a frequency ordering of letters, and selection or rejection of a particular letter is based on that letter's relative order in the table with respect to the previous letter. It thus gives a statistical advantage, not an absolute one, to the lower frequency letters, since many words differ only in high frequency vowels (e.g., COMPUTE, COMPETE, COMPOTE). The relative order feature adds a randomizing aspect to selection that permits inclusion of occasional high frequency letters.

The frequency ordering used is taken from tables in Pratt(1). Different word samples will yield slightly different orderings, but the cipher resolution is not sensitive to minor orderings. The Pratt ordering is:

ETAONRISHDLFCMUGYPWBVKXJQZ

Algorithm

The algorithm is: "If a source word is longer than six letters, select the first letter and subsequent letters of lesser or equal ordering than the prior letter, and continue the process recursively until six letters remain. Words of six letters or less are reproduced in full and filled out with null symbols, where necessary, until a total of six characters is reached."(4)

Several examples will illustrate the system. Omitted letters are shown bracketed, and successive cycles are shown by arrows.

1. B[I]B[LIO]G[RA]P[H]ER—►BBGPER
2. I[N]F[O]RM[AT]I[O]N—►IFRMIN
3. SH[A]K[E]SP[E]AR[E]—►SHK[S]PAR—►SHKPAR
4. SMITH—►SMITH □
5. K[IN]G[S]F[O]RD[-S]M[IT]H—►K[G]FRDMH—►
   KFRDMH
6. K[R]ISH[NA]M[O]OR[T]H[I]—►K[I]SHM[O]RH—►
   KSHMRH

In some very rare cases, an emerging cipher may have more than six letters in descending sequence, so that it will not decompose further. In such cases the final letters are eliminated until six remain.

Most words, however, will reduce in one or two cycles. In a test of 55,000 words only one was found requiring four cycles. A few extreme cases do exist, however: the longest ever found required six cycles:

7. AN[T]ID[I]S[E]S[T]AB[LI]SHM[E]N[T]ARI[A]NISM—►
   ANID[S]S[A]B[S]HM[NA]RI[N]ISM—►
   ANID[S]B[H]M[R]IISM—►
   ANIDB[MI]ISM—►
   ANIDB[I]SM—►
   ANIDB[S]M—►
   ANIDBM

Only slightly shorter, the longest word in Shakespeare's works (*Love's Labour's Lost* V.i) reduces in three recursions:

8. H[O]N[O]RIF[I]C[A]B[I]L[IT]U[DIN]I[T]A[T]IB[US] —►
   H[N]RIFCB[L]U[IA]IB —►
   H[R]IFCB[UI]B —►
   HIFCBB

Even Mary Poppins' sesquipedalian ecphonesis crumbles to six letters in three recursions:

9. SUP[E]RC[A]L[I]F[RA]G[I]L[I]S[T]IC[E]X[PIA]L[I]
   D[O]C[O]U[S] —►
   SUPRC[L]FG[LSI]CX[LD]CU —►
   SUP[CF]G[C]X[C]U —►
   SUPGXU

The prime advantages of the method are its computational simplicity and its resolution. The elimination requires only table lookup and no multiplications; and the compression is readily done manually. The resolution is apparently as good as one can get with a selected letter compression code. It effectively flattens the high portions of the letter frequency curve, though unlike a randomizing code it cannot totally equalize the distribution. The resolution, however, is quite good. Specifically in a test of 4862 words (chosen from the secretary's handbook *20,000 Words*), only thirty of the six-letter ciphers (about 0.61%) were non-unique and of the non-unique ciphers all were simple pairs except for one instance of three occurrences. The method compresses quickly: since all non-initial letters have a .5 probability of being retained, the expected length, *L*, of an n letter word after r recursions is:

$$L = 1 + \frac{n-1}{2^r}$$

This indicates that a 43-letter word may be expected to compress to six letters in three recursions.

THE SOUNDEX CODE

The widely used Soundex code(5), has been attributed to Remington Rand. It is a phonetic code that tends to create identical codes from similar sounding names. It is useful for name searching under conditions of uncertainty of spelling, such as occurs in the airline reservation problem where it is often required to match a telephoned name in a machine file. The code has five steps:

1. Retain first letter of name as first letter of code.
2. Eliminate vowels, W, H, and Y.
3. Eliminate the second consonant of a double consonant pair.
4. Replace the following letters by numbers:

B,P,F,V,	1
C,G,J,K,Q,S,X,Z,SC,CH,SCH,CK	2
D,T	3
L	4

M,N	5
R	6

5. Take the first three or four symbols, and add zeros if insufficient phonetic sounds.
   The example below illustrates the process:
   JOHNSEN $\rightarrow$ JNSN $\rightarrow$ J525 $\rightarrow$ J52
   JOHNSON $\rightarrow$ JNSN $\rightarrow$ J525 $\rightarrow$ J52
   JOHNSTON $\rightarrow$ JNSTN $\rightarrow$ J5235 $\rightarrow$ J52
   JOHNSTONE $\rightarrow$ JNSTN $\rightarrow$ J5235 $\rightarrow$ J52

CONCLUSION

Historically, compression techniques for word coding have been designed for both encoding and use by humans. Here described are some codes requiring computers for practical encoding usable by humans. As files grow larger and directory code generation becomes more demanding, it is likely that alphanumeric concessions to human readers will be eliminated in favor of more efficient use of the code space. The codes presented here, however, appear quite useful for many present applications in information retrieval.

ACKNOWLEDGMENT

This work has been performed as a part of the development of NELINET (The New England Library Information Network) under contract to the New England Board of Higher Education; and has been supported by the Council on Library Resources, Inc., under Contract CLR-385.

REFERENCES

1. Pratt, Fletcher: *Secret and Urgent, the Story of Codes and Ciphers* (Garden City, New York: Blue Ribbon Books, 1939).
2. Gaines, Helen Fouché: *Cryptanalysis* (New York: Dover, 1956).
3. Kahn, David: *The Codebreakers* (New York: Macmillan Company, 1967).
4. Nugent, William R.: Vegh, Alexander: *Automatic Word Coding Techniques for Computer Language Processing.* Report RADC-TDR-62-13 February, 1962. Volume I: AD-272-401. Volume II: AD-272-402.
5. Wright, M. A.: "Mechanizing a Large Index; Appendix: The Soundex Code," *The Computer Journal,* 3 (July 1960), p. 33.