AN ALGORITHM FOR DIFFERENTIATING NATURAL LANGUAGE WORDS FROM NONSENSE WORDS
(UN ALGORITHME POUR DIFFERENCIER LES MOTS EN LANGAGE NATUREL DES SUITES ININTELLIGIBLES DE CARACTERES)

Frank T. Dolan
School of Library and Information Science
The University of Western Ontario
London, Ontario
N6A 5B9

ABSTRACT

A heuristic algorithm is presented which enables a digital computer to seemingly transcend the syntactical level of symbol manipulation and enter the semantic level. The result is a program which can differentiate natural language (English) words from nonsense words. The device which makes such differentiation possible is a probability matrix which is derived from a list of basic English vocabulary. This table gives the probabilities of an alphabetic character (A through Z including blank) occurring as the 1st, 2nd, . . . nth character in an English word. The operation of this program is demonstrated by feeding a cryptogram of unspecified type into the machine. The computer starts decoding the cryptogram in all possible ways checking as it goes on the probability of each result being English. When the probability of one of its products being English excedes the probability of all the others being English by a specifiable magnitude, the computer stops using the other decoding subroutines and proceeds with the "right" one, displaying only the English result. (L'auteur présente un algorithm heuristique qui permet à ordinateur digital de transcender, en apparence, le niveau syntaxique de la manipulation des symboles et de pénétrer dans l'univers sémantique. Il en résulte un logiciel capable de différencier les mots anglais, en langage naturel, des suites inintelligibles de caractères. L'expédient permettant cette différenciation est une matrice de probabilité, dérivée d'un vocabulaire anglais de base. Cette table donne les probabilités, pour un caractère alphabétique (de A à Z incluant les espaces en blance), d'apparaître en tout que 1er, 2eme, . . . nième caractère, dans un mot anglais. Le fonctionnement de ce logiciel est apparent lorsqu'ou soumet à la machine un cryptogramme d'un type indéterminé. L'ordinateur commence à décoder le cryptogramme de toutes les manières possibles, vérifiant, au fur et à mesure, la probabilité que chacun des résultats soit acceptable en anglais. Lorsque la probabilité que l'une des solutions soit acceptable en anglais, excède, par une grandeur prédeterminée, la probabilité que toutes les autres solutions soient acceptables en anglais, l'ordinateur cesse d'utiliser toutes les autres sous-programmes et continue d'opérer avec la routine appropriée, affichant seulement le résultat "anglais".)

BASIC ENGLISH WORDS

The basis for the algorithm described in this paper is the Basic English Word List of Richards and Ogden (Richards, 1943). A sample of this list is reproduced in Figure 1.

Figure 1

Indicative Sample of Basic English Word List

OPERATIONS	THINGS			QUALITIES	
	General		Picturable	General	Opposities
come	account	metal	angle	able	awake
be	answer	mother	basin	cheap	dead
after	back	noise	boat	dependent	future
off	body	ornament	brick	free	narrow
of	canvas	person	carriage	healthy	simple
some	committee	powder	coat	material	
but	cork	pull	drain	physical	
ever	current	reaction	feather	red	
together	design	rest	garden	smooth	
so	disgust	run	heart	tired	
	education	sense	knee		
	expert	silver	match		
	fire	smoke	nose		
	fruit	space	pin		
	harbor	story	rail		
	humor	system	screw		
	interest	thunder	sock		
	knowledge	twist	stick		
	lift	wash	thumb		
	man	wine	umbrella		

FREQUENCY TABLE

The first step was to input the total list into a machine-readable file. The second step was to use these 850 words to produce the frequency table shown in Figure 2.

Figure 2

Frequency of character occurrence by position within word for Basic English Words

POSITION WITHIN WORD

CHARACTER	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BL	0	2	16	89	241	195	126	80	50	33	9	5	2	1	1
A	44	117	90	36	22	17	9	4	0	1	0	0	0	0	0
B	66	3	8	7	2	2	1	0	1	0	0	0	0	0	0
C	68	10	28	32	15	11	5	4	2	0	0	0	0	0	0
D	43	6	19	45	14	8	4	0	0	0	0	0	0	0	0
E	26	126	68	109	113	47	27	18	8	1	2	0	0	1	0
F	49	7	10	8	3	1	1	0	0	0	0	0	0	0	0
G	22	4	26	20	18	6	13	5	0	0	0	0	0	0	0
H	32	52	3	20	28	15	5	0	0	0	0	0	0	0	0
I	18	87	68	36	35	25	16	5	3	2	0	1	0	0	0
J	6	0	1	0	0	0	0	0	0	0	0	0	0	0	0
K	11	3	5	29	14	1	0	0	0	0	0	0	0	0	0
L	34	38	49	45	28	13	8	6	2	1	0	0	0	0	0
M	41	12	31	26	5	5	4	1	0	1	0	0	0	0	0
N	25	30	50	50	33	37	17	17	14	5	3	3	0	0	0
O	23	149	74	23	25	8	12	10	4	3	2	0	0	0	0
P	59	14	23	31	7	3	1	0	0	0	0	0	0	0	0
Q	5	2	2	1	0	0	0	0	0	0	0	0	0	0	0
R	39	76	83	44	45	36	12	8	0	0	0	0	0	0	0
S	122	3	52	42	22	16	8	5	2	0	0	0	0	0	0
T	53	33	55	85	50	40	26	9	10	4	2	0	1	0	0
U	5	56	36	25	10	4	1	4	0	0	0	0	0	0	0
V	7	6	14	8	1	3	0	0	0	0	0	0	1	0	0
W	46	5	14	12	1	4	0	1	0	0	0	0	0	0	0
X	0	6	6	0	0	0	1	0	0	0	0	0	0	0	0
Y	6	3	16	9	10	5	9	4	5	0	0	0	0	0	0
Z	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

As this table shows, 44 of the basic English words start with the letter "A". One hundred and seventeen of the basic English words have "A" as their second letter. Ninety of these words have "A" as their third letter and so on through A and the other letters. The first entry in the table shows the frequency of a blank space immediately following a letter. This serves as a counter on various word lengths. For instance, there are 2 occurrences of such blanks in the 2nd letter position which means that there are two one-letter words in the total 850.

PROBABILITY TABLE

The frequency table can be easily converted into a probability table by dividing the actual number of occurrences in each cell by the total number of words, i.e. 850. Figure 3 shows the results after such a conversion has taken place.

The probabilities shown in this table have been rounded off to three decimal places. In the table, the probability of "A" being in the 1st letter position of a basic English word is shown as .052 . In the probability file used by the program, this is expressed as 5.17648E-2, where "E-2" reads "times 10 to the -2 power". Probabilities shown in the table as .000 have been replaced in the probability file by 1.0E-10. This has been done because a letter which does not occur in a certain position in the basic English word list may, of course, occur in extended English. For example, "X" and "Z" do not occur in letter position 1 of any of the Basic English words but each does occur in extended English words such as Xenon and Zeal. Since only an impossible event should have a probability of zero, a probability of 1.0E-10 has been arbitrarily assigned to such cases. Ultimately cumulative probabilities will be calculated by multiplying individual probabilities. In a case where one such probability was zero, the cumulative probability would also be reduced to zero. The assignment of a probability of 1.0E-10 to such cases elminates this possibility while preserving the low probability of such an occurrence.

CRYPTOGRAM ENCODING AND DECODING

The next step was to generate a series of cryptograms as test data for the discrimination algorithm. Figure 4 shows the tables for encoding and decoding these cryptograms.

Figure 3

Probability of character occurrence (including blank, BL) by position within word for Basic English words

POSITION WITHIN WORD

CHARACTER	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BL	.000	.002	.019	.105	.283	.229	.148	.094	.059	.039	.011	.006	.002	.001	.001
A	.052	.136	.106	.042	.026	.020	.011	.005	.000	.001	.000	.000	.000	.000	.000
B	.078	.003	.009	.008	.002	.002	.001	.000	.001	.000	.000	.000	.000	.000	.000
C	.080	.012	.033	.038	.018	.013	.006	.005	.002	.000	.000	.000	.000	.000	.000
D	.051	.007	.022	.053	.016	.009	.005	.000	.000	.000	.000	.000	.000	.000	.000
E	.031	.148	.080	.128	.133	.055	.032	.021	.009	.001	.002	.000	.000	.001	.000
F	.058	.008	.012	.009	.003	.001	.001	.000	.000	.000	.000	.000	.000	.000	.000
G	.026	.005	.031	.023	.021	.007	.015	.006	.000	.000	.000	.000	.000	.000	.000
H	.038	.061	.003	.023	.033	.018	.006	.000	.000	.000	.000	.000	.000	.000	.000
I	.021	.102	.080	.042	.041	.029	.019	.006	.003	.002	.000	.001	.000	.000	.000
J	.007	.000	.002	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
K	.013	.003	.006	.034	.016	.001	.000	.000	.000	.000	.000	.000	.000	.000	.000
L	.040	.045	.058	.052	.033	.015	.009	.007	.002	.001	.000	.000	.000	.000	.000
M	.048	.014	.036	.031	.006	.006	.005	.001	.000	.001	.000	.000	.000	.000	.000
N	.029	.035	.059	.059	.039	.043	.020	.020	.016	.006	.003	.003	.000	.000	.000
O	.027	.175	.087	.027	.029	.009	.014	.012	.005	.003	.002	.000	.000	.000	.000
P	.069	.016	.027	.036	.008	.003	.001	.000	.000	.000	.000	.000	.000	.000	.000
Q	.006	.002	.002	.001	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
R	.046	.089	.098	.052	.053	.042	.014	.009	.000	.000	.000	.000	.000	.000	.000
S	.143	.003	.061	.049	.026	.019	.009	.006	.002	.000	.000	.000	.000	.000	.000
T	.062	.039	.065	.100	.059	.047	.031	.011	.012	.005	.002	.000	.001	.000	.000
U	.006	.066	.042	.029	.012	.005	.001	.005	.000	.000	.000	.000	.000	.000	.000
V	.008	.007	.016	.009	.001	.003	.000	.000	.000	.000	.000	.000	.000	.000	.000
W	.054	.006	.016	.014	.001	.005	.000	.001	.000	.000	.000	.000	.000	.000	.000
X	.000	.007	.007	.000	.000	.000	.001	.000	.000	.000	.000	.000	.000	.000	.000
Y	.007	.003	.019	.011	.012	.006	.011	.005	.006	.000	.000	.000	.000	.000	.000
Z	.000	.000	.001	.000	.001	.000	.001	.000	.000	.000	.000	.000	.000	.000	.000

Figure 4

Cryptogram Encoding (left) and Decoding (right) Tables

Encoding — CRYPTOGRAM TYPE

CHARACTER	1	2	3	4	5	6	7	8	9	10
	BL	BL	BL	BL	BL	BL	BL	BL	BL	BL
A	L	H	X	Z	W	Y	W	G	K	D
B	M	K	L	P	M	M	L	Q	T	A
C	P	V	I	S	B	A	M	A	E	F
D	D	G	N	T	C	U	A	V	G	B
E	K	B	O	F	F	V	F	N	X	T
F	J	I	P	D	R	F	I	L	D	I
G	Z	L	M	C	T	J	E	K	R	U
H	O	U	U	W	Z	G	C	M	J	W
I	F	Q	K	X	I	O	T	R	A	Q
J	A	E	Z	M	X	P	B	F	L	J
K	B	N	H	Q	H	N	N	S	Y	Y
L	S	F	F	A	L	E	V	B	Q	V
M	N	M	T	J	E	L	O	H	C	Z
N	U	Z	Y	V	J	Z	K	T	S	H
O	E	X	W	I	V	X	U	X	P	G
P	T	A	S	G	A	W	Q	P	M	C
Q	V	J	J	H	K	B	J	C	U	S
R	I	Y	B	K	S	K	Z	W	O	K
S	C	C	R	N	G	I	D	U	Z	N
T	H	T	G	O	N	D	P	Z	B	L
U	X	S	Q	Y	U	R	G	E	W	R
V	G	R	C	B	Y	C	S	D	F	X
W	N	O	E	U	Q	Q	R	J	H	M
X	Y	D	V	E	P	H	X	O	I	P
Y	Q	W	D	L	O	S	Y	I	V	O
Z	R	P	A	R	D	T	H	Y	N	E

Decoding — CRYPTOGRAM TYPE

CHARACTER	1	2	3	4	5	6	7	8	9	10
	BL	BL	BL	BL	BL	BL	BL	BL	BL	BL
A	J	P	Z	L	P	C	D	C	I	B
B	K	E	R	V	C	Q	J	L	T	D
C	S	S	V	G	D	V	H	Q	M	P
D	D	X	Y	F	Z	T	S	V	F	A
E	O	J	W	X	M	L	G	U	C	Z
F	I	L	L	E	E	F	E	J	V	C
G	V	D	T	P	S	H	U	A	D	O
H	T	A	K	Q	K	X	Z	M	W	N
I	R	F	C	O	I	S	F	Y	X	F
J	F	Q	Q	M	N	G	Q	W	H	J
K	E	B	I	R	Q	R	N	G	A	R
L	A	G	B	Y	L	M	B	F	J	T
M	B	M	G	J	B	B	C	H	P	W
N	W	K	D	S	T	K	K	E	Z	S
O	H	W	E	T	Y	I	M	X	R	Y
P	C	Z	F	B	X	J	T	P	O	X
Q	Y	I	U	K	W	W	P	B	L	I
R	Z	V	S	Z	F	U	W	I	G	U
S	L	U	P	C	R	Y	V	K	N	Q
T	P	T	M	D	G	Z	I	N	B	E
U	N	H	H	W	U	D	O	S	Q	G
V	Q	C	X	N	O	E	L	D	Y	L
W	M	Y	O	H	A	P	A	R	U	H
X	U	O	A	I	J	O	X	O	E	V
Y	X	R	N	U	V	A	Y	Z	K	K
Z	G	N	J	A	H	N	R	T	S	M

DIFFERENTIATING NATURAL LANGUAGE

Ten types of cryptogram were generated by simply drawing the letters A through Z from a container without replacement and assigning the letter chosen successively through the alphabet. For example, to establish a type 1 cryptogram, a letter was drawn and assigned to A. This letter happened to be "L". The next letter drawn, "M", was assigned to B, and so through the alphabet. The twenty-six letters were then put back in the container, thoroughly mixed and successively redrawn and reassigned to create the additional cryptogram types. The decoding table was prepared by rearranging the encoding table so that any entry in the former became an argument in the latter.

THE DISCRIMINATION ALGORITHM

The program receives a cryptogram of unknown type and a probability factor from the terminal. It then procedes character by character to decode the cryptogram in all ten possible ways. As it proceeds, it generates a cumulative probability for each cryptogram type. After it finishes decoding the character for each type, it checks to see if the largest cumulative probability is greater than the second largest cumulative probability factor multiplied by the inputted probability factor. If so, it stops decoding, identifies the type associated with the most probable product and prints out the result using that type for decoding.

SAMPLE OUTPUT

```
WHAT IS YOUR CRYPTOGRAM; 'STOP' TO END
 ?L NEID HE HOK NFCK FC CXJJFPFKUH
WHAT IS YOUR PROBABILITY FACTOR ?10
CHARACTER COUNT= 1
SORTING
 7.05882E-3     4
 7.05882E-3     9
 2.58824E-2     2
 0.04           5
 4.82353E-2     6
 5.17647E-2     1
 5.76471E-2     8
 6.23529E-2     10
 7.76471E-2     3
 7.76471E-2     7
CHECKING
```

```
CHARACTER COUNT= 2
SORTING
  1.66090E-5    4
  1.66090E-5    9
  6.08997E-5    2
  9.41176E-5    5
  1.13495E-4    6
  1.21799E-4    1
  1.35640E-4    8
  1.46713E-4    10
  1.82699E-4    3
  1.82699E-4    7
CHECKING
CHARACTER COUNT= 3
SORTING
  1.66090E-15   9
  7.88113E-7    2
  1.46876E-6    6
  2.36434E-6    7
  2.38388E-6    4
  4.14899E-6    8
  5.86851E-6    5
  6.59149E-6    1
  9.24242E-6    3
  2.10576E-5    10
CHECKING
CHARACTER COUNT= 4
SORTING
  1.95400E-17   9
  7.88113E-17   2
  2.10576E-15   10
  1.11263E-8    7
  1.68274E-8    4
  5.43672E-8    3
  6.56620E-8    6
  8.28496E-8    5
  2.73345E-7    8
  1.15545E-6    1
CHECKING
CHARACTER COUNT= 5
SORTING
  1.37929E-19   9
  9.27192E-19   2
  2.47736E-17   10
  1.30898E-10   7
  1.46497E-9    4
  1.79092E-9    3
  4.01697E-9    6
  5.14532E-9    8
  6.62797E-9    5
  1.12826E-7    1
CHECKING
A WORD TO THE WISE IS SUFFICIENT
WHAT IS YOUR CRYPTOGRAM; 'STOP' TO END
 ?STOP
```

Here we see a trace of what actually happens in the computer followed by the proper decoding of the cryptogram. Note the column of probabilities followed by the column of associated cryptogram types. The cryptogram we submitted here was a type 1 cryptogram. The computer, of course, did not "know" this but had to start decoding it character by character calculating the probability of every product until the highest probability was 10 times (Note that we submitted a probability factor =10) the next highest. Note that type 1 started out with middle range probability but migrated rapidly up the table to take the lead after only 4 characters had been decoded. By character 5, the probability that it was a type 1 cryptogram was more than 10 times greater than the probability that it was a type 5 cryptogram (its closest competitor) so the computer set type = 1 and decoded the cryptogram properly.

CONCLUSIONS

This demonstration shows the power that can be built into a relatively simple algorithm using an extremely small reference base of 850 Basic English words. It is hypothesized that the algorithm could be made even more powerful by adding more sophistication to the program, i.e. looking at the occurrence patterns of groups of characters rather than at single characters. Also its discrimination power should be enhanced by extending the reference base possibly to a full dictionary.

One refinement that is underway is to calculate the probabilities of each word in the Basic English word list and do a statistical analysis of the distribution of these probabilities. Given this distribution, it should be possible to determine within confidence limits the probability of any given word being English. Furthermore, the principles outlined here should apply to any language, natural or artificial. If the patterns lexical, phonemic, etc. of any language can be validly quantified, it should be possible for the computer to recognize these patterns and discriminate accordingly within the derived probability limits.

REFERENCES

RICHARDS, I. A. 1943 Basic English and Its Uses. New York, W. W. Norton & Company, Inc. 143 p.