〈経営研究　第13巻　第2号　平成11年12月〉

英書19冊の15次エントロピーの計算

横　井　右　門

1955年にバーナードが英語について8次エントロピーが2.35であると計算した。当時と違いコンピュータの性能は格段に向上したのでもう一度大量テキスト・データにもとづいて計算しなおしてみようというのが本論の目的である。

キーワード

エントロピー　entropy

冗長度　redundancy

序論

シャノンは熱力学における無秩序の度合いであるエントロピーを援用し，情報源に存在する不確実さの度合いをエントロピーと呼んだ。(1)

南によれば，バーナードG. A. Barnardは1955年にStatistical Calculation of Word Entropies for Four European Languagesにおいて，つぎのようにエントロピーを計算している。(2)

	英語	仏語	独語	スペイン語
1次	4.124	3.98	4.10	4.015
2次	3.56			
3次	3.3			
8次	2.35			

翌1956年に当時ソ連のヤグロムは英語，フランス語，ドイツ語のエントロピーがそれぞれ1.242，1.200，1.233単位であると発表している。2を底とする対数に換算するために10を底とする2の対数0.3010で割ると，それぞれ4.126，3.99，4.10となり，ほとんど一致する。さらにヤグロムはシャノンが英語の2次および3次エントロピーを1.075および0.993と計算したとしているが[3]，同じ操作を施すと3.57，3.3となり，バーナードの計算に一致する。同じアメリカ人が書いたクランシーのThe Hunt for Red Octoberとパーカーの A Catskill Eagleの1次エントロピーが0.05も違うのに、バーナードとヤグロムの計算がここまで似ているのは極めて興味深い。

また堀はつぎのように言っている。[4]

何しろ，前にみたように，文字のつながり方の制約は，三つや四つどころでなく，もっともっと長い文字の連鎖まで及ぶことが明らかなのだから。たぶん，少なくとも14か15の相つづく文字の列のすべての可能なものを数えあげ，それらの頻度分布をしらべてエントロピーを計算しなければ，実際の値に近い値は求まらないだろう。しかしそれは大変な仕事で，まだ誰もやっていない。ある人が英語の中に出てくる相つづく八つの文字のすべての列の出現頻度をしらべてエントロピーを計算したのが今までの最高記録である。その値は2.35であったが，実際はこれよりもっと小さいだろう。あとは大ざっぱに見当をつけるよりほかないのだが，まず1.3ぐらいだろうというのが，今のところの定説になっている。

情報理論の多くのテキストが冗長度について次のように表現していることが多い。

「0.75 程度であると推定されている」[5]

これによれば,「実際の英語」のエントロピーは 1.175 ということになる。冗長度は相対エントロピー，すなわち「実際の英語」のエントロピーをカオスのエントロピーで割ったものを 1 から引いたものである。したがって次の 1 次方程式を解けばよい。

$$1-\frac{x}{4.7}=0.75$$

「英語の実際の冗長度はこの値（二分の一）よりもかなり大きいものと推測されている」[2]

このような少し曖昧ともとれる表現が散見されるのは，ピアースによればシャノンが巧妙な実験によって示したとされる 0.6 ～ 1.3 ビット[6] が根拠になっていると思われる。

本報告はより正確な推測に資するために，実験ではなく最近の英文をデータ処理することにより，できるだけ高次の 15 次までのエントロピーを計算する。

データ作成

つぎの 19 冊のテキストを使用した。著者名の下に略号と書名を列記する。

Tom Clancy 著

C1　The Hunt for Red October：Harper Collins，1993

C2　OP Center：Op-Center：Berkley Novel，1995

Frederick Forsyth 著

F1　The Day of the Jackal：Bantam Books，1995

F2　The Dogs of War：Bantam Books，1995

F3　The Odessa File：Bantam Books，1973

F4　The Shepherd：Corgi Books，1990

Akira Kohchi 著

K1　Why I survived A-Bomb：Institute for Historical Review，1989

Anne McCaffrey 著

M1　The Crystal Singer：Corgi Books，1991

M2　Crystal Line：Del Rey Books，1992

Robert B. Parker 著

P1　A Catskill Eagle：Dell，1985

P2　Pastime：Berkley Novel，1992

P3　God Save the Child：Penguin Books，1977

P4　The Godwulf Manuscript：Dell，1987

A. J. Quinnell 著

Q1　Man on fire：Orion，1994

Q2　In the Name of the Father：Signet Novel，1987

Q3　The Blue Ring

Q4　Message from Hell：Orion，1996

Jostein Gaarder 著

GA　Sophie's World：Berkley Books，1996

H. G. Wells 著

WE A Short History of the World，Collins，1953

以上の19冊を約2年半かけて手作業で入力した。スキャナー等による入力については岡田毅氏の先見的な業績があるが，あえて手作業を選んだ。かりに正読率99パーセントのシステムが入手できたとしても，ペーパーバックの1ページは概ね2000バイトであるから1ページあたり20字の間違いがあることになる。また全角文字，ファイル・エンド等の文字が作られたりすると，プログラム実行時のトラブルの原因になってしまう。それならば，正確なタッチ・メソッドによるタイピングが可能な人間が慎重に入力した方がよい。

著作権の問題については研究のためならばコピーしても構わないという見解もあるが[8]，本報告については，例文はK1から引用することにした。

「英語」とは何か

たとえばC1とP1の1次エントロピーを比較すると，4.080464と4.028679になり，ずいぶん違う。前川氏の「文章を科学する」において「English」とされている文献の集まりは，読む人もいれば読まない人もいるであろうし，同書において使用された日本語のテキストの集まりは，池波正太郎や藤沢周平等の時代物しか読まない人にとっては違和感があるはずである。

そこでこの報告では，「英語」についてエントロピーを計算することを諦め，個々のテキストごとにエントロピーを計算することにした。

入力の仕方はつぎの通りである。

①アスキー96文字と改行キーだけで入力する。

②パラグラフの始まりはスペース2個とする。

③右寄せはしない。すなわちright justificationはしない。

④ダッシュはハイフン2個で置き換える。

⑤ページ番号は左寄せとし，その前後に改行キーを3個置く。したがってページ番号の行の前後に空白行が2行ずつできる。

⑥タブ・キーは使わない。処理系によりタブの位置が異なるからである。

⑦イタリックは無視して普通の文字にする。

⑧chapterごとに1テキスト・ファイルを作る。

⑨字下げindentionは原文のレイアウトに合わせ，適当な数のスペースをタイプする。実はこれがエントロピーを計算するにあたり，意外な影響がある。

エントロピー計算プログラム

処理系は次のとおりである。

ハードウエア Gateway Pentium Ⅱ 400MHz メモリ 64 メガバイト

ハードディスク　4 ギガバイト

言語 VisualC^{++} 4.0

本報告ではアルファベット26文字とスペースの計27文字だけを処理した。この処理系では配列を使って計算した場合3次エントロピーまでしか計算できない。4次エントロピーを計算しようとして27の4乗の配列を指定すると，オブジェクト・モジュールまでは作るが，実行モジュールは作成しない。

さらに予想していたことではあるが，今回のエントロピー計算の結果，付随して出てきたデータであり，論理の展開としては，順序が逆になってしまうが，実際に発生する文字列について言及しておく。1次エントロピーは文字列の長さが1であり，27文字はよほど短いテキスト・データでない限りどの文字も必ず発生する。27要素の1次元配列の占有率は100パーセントにな

る。二次元配列を定義すれば要素の数は27の2乗の729になるが，実際には610しか発生しない。具体的にいうと，C1においてanは10732個発生しているが，bgは発生していない。610は，729に対して83.7パーセントである。27の3乗は19683であるが，実際に発生した3連字の種類は5841で，19683に対して29.7パーセントである。以下，実際に発生した4連字，5連字，6連字，7連字の組み合わせはそれぞれ26459，78459，171162，283304であり，その論理的可能性に対する比率は，それぞれ4.98パーセント，0.55パーセント，0.04パーセント，0.003パーセントである。4連字の場合でも配列のために用意したメモリー空間のわずか5パーセントしか使わないのでは，経営情報システムを開発した経験からは，まったくメモリーの無駄遣いだと言うことになる。

そこで配列を使わずに，すべてハード･ディスク上で処理することにした。手続きは次の3段階である。15次エントロピーを例として説明する。(付録参照)

第1段階（プログラムcreasy15. c)

1　1文字読んで配列内で1文字シフトする
2　英字とスペース以外は無視する
3　大文字は小文字にする
4　改行はスペースにする
5　15次エントロピーの場合は15文字と改行キー（CRとLFで2バイト）の3バイトのレコードを出力する。
6　次の1文字を読む
7　ファイルの総字数を1ファイルとして出力する

第2段階

これを昇順で整列sortする。

第3段階（プログラム pinta15. c)

1　第1段階の総字数を読み，確率計算のための分母とする

2　同一文字列の場合1を加算する

3　文字列が変わったら確率 p を計算し，情報量 plogp を計算し加算する

4　エントロピーを出力する

ブロック化の問題

K1 の 91 ページに次の文がある。

It appeared as if the entire air sparked.

2次エントロピーを計算するにあたり，つぎの2通りの方法が考えられる。第一の方法は次のように2連字の文字列を生成することである。

[It] [a] [pp] [ea] [re] [d] [as] [i] [f] [th] [e] [en] [ti] [re] [a] [ir] [s] [pa] [rk] [ed]

第2の方法はつぎのように文字列を生成することである。

[It] [t] [a] [ap] [pp] [pe] [ea] [ar] [re] [ed] [d] [a] [as] [s] [i] [if] [f] [t] [th] [he] [e] [e] [en] [nt] [ti] [ir] [re] [e] [a] [ai] [ir] [r] [s] [sp] [pa] [ar] [rk] [ke] [ed] [d]

第1の方法をブロック化という[5]。しかし，これによってエントロピーを計算するのであれば，[t] [ap] [pe] [ar] [ed] 等の発生確率が計算できないことになる。また，この文の前方でスペースが1個余計にあるかないかで発生確率の計算対象になる文字列が選択されてしまうことになる。そこで，第2の方法を選択することにした。とりあえず非ブロック化と呼んでおく。情報理論は，ある地点から別の地点にいかに効率よく情報を伝達するかという発想であるからブロック化もやむをえないと思われる。しかしエントロピーの概念は情報理論以外に言語学の世界でも有用な概念とされている。[7]

計算結果

	C1	C2	F1	F2	F3
字数	970, 567	537, 312	771, 683	810, 586	596, 877
平均語長	4. 414206	4. 312595	4. 435900	4. 413581	4. 325885
標準偏差	2. 392379	2. 369556	2. 414696	2. 368385	2. 333826
最大語長	27	35	22	24	35
H01	4. 080464	4. 058987	4. 063707	4. 068114	4. 063637
H02	3. 743243	3. 710724	3. 695529	3. 704562	3. 705528
H03	3. 414249	3. 383596	3. 366309	3. 379146	3. 373736
H04	3. 097624	3. 071304	3. 059774	3. 069669	3. 061388
H05	2. 824913	2. 796666	2. 791683	2. 800516	2. 787476
H06	2. 592527	2. 556590	2. 558452	2. 567902	2. 549024
H07	2. 383729	2. 339179	2. 350169	2. 359863	2. 335465
H08	2. 195781	2. 144437	2. 163240	2. 172658	2. 144504
H09	2. 025198	1. 968584	1. 993765	2. 002778	1. 971894
H10	1. 870136	1. 810944	1. 840903	1. 849168	1. 816953
H11	1. 730735	1. 670632	1. 703612	1. 711402	1. 678705
H12	1. 605754	1. 546156	1. 580527	1. 587881	1. 555438
H13	1. 494354	1. 436241	1. 470817	1. 477742	1. 446063
H14	1. 395350	1. 339286	1. 373321	1. 379690	1. 349280
H15	1. 307246	1. 253437	1. 286567	1. 292403	1. 263380

	F4	K1	M1	M2	P1
字数	64, 003	371, 522	644, 252	511, 075	404, 989
平均語長	4. 270953	4. 757866	4. 623770	4. 414708	3. 989992
標準偏差	2. 401242	2. 633331	2. 590444	2. 438119	2. 105130
最大語長	35	27	27	31	24
H01	4. 077524	4. 105486	4. 090534	4. 071662	4. 028679
H02	3. 714410	3. 752478	3. 725992	3. 715240	3. 674845
H03	3. 345568	3. 439573	3. 391644	3. 377660	3. 321701
H04	2. 983058	3. 128984	3. 074152	3. 059975	3. 001792
H05	2. 650899	2. 842895	2. 800287	2. 784370	2. 722758
H06	2. 358815	2. 581869	2. 563205	2. 545010	2. 481822
H07	2. 107253	2. 345396	2. 350100	2. 329624	2. 264267
H08	1. 892638	2. 134097	2. 158622	2. 136166	2. 072986
H09	1. 710381	1. 946667	1. 984904	1. 961309	1. 901181
H10	1. 555489	1. 781914	1. 828577	1. 804314	1. 750305
H11	1. 423540	1. 638262	1. 688669	1. 664511	1. 616198
H12	1. 310327	1. 512593	1. 564224	1. 540635	1. 497783
H13	1. 212815	1. 402714	1. 454025	1. 431184	1. 393471
H14	1. 128336	1. 306482	1. 356610	1. 334667	1. 301560
H15	1. 054509	1. 219384	1. 270342	1. 249226	1. 219916

	P2	P3	P4	Q1	Q2
字数	287, 287	298, 069	304, 275	547, 589	638, 157
平均語長	3. 957715	3. 981495	3. 972066	4. 348522	4. 312332
標準偏差	2. 079894	2. 183470	2. 177281	2. 381582	2. 298410
最大語長	29	34	34	27	27
H01	4. 036607	4. 053653	4. 054947	4. 063091	4. 073119
H02	3. 682573	3. 700960	3. 700537	3. 704640	3. 713228
H03	3. 323749	3. 351761	3. 350905	3. 366370	3. 381943
H04	2. 997496	3. 029806	3. 028415	3. 051857	3. 068014
H05	2. 712235	2. 745430	2. 745693	2. 777384	2. 793411
H06	2. 465206	2. 494746	2. 496762	2. 539175	2. 554875
H07	2. 243048	2. 269285	2. 273822	2. 326873	2. 342277
H08	2. 048125	2. 069805	2. 075818	2. 137097	2. 152050
H09	1. 873878	1. 891771	1. 898199	1. 965071	1. 979701
H10	1. 721260	1. 735434	1. 741373	1. 810355	1. 824531
H11	1. 586318	1. 597718	1. 603143	1. 671985	1. 685551
H12	1. 467813	1. 477318	1. 481935	1. 548762	1. 561938
H13	1. 363587	1. 371623	1. 375598	1. 439382	1. 452331
H14	1. 272110	1. 278915	1. 282204	1. 342514	1. 355366
H15	1. 159259	1. 197006	1. 199795	1. 256626	1. 269268

	Q3	Q4	GA	WE	Q0
字数	622, 674	468, 054	964, 734	712, 486	10, 853
平均語長	4. 231167	4. 152141	4. 370858	4. 822602	4. 302100
標準偏差	2. 209136	2. 234328	2. 431823	2. 702675	2. 455804
最大語長	20	20	31	27	22
H01	4. 060667	4. 059951	4. 054536	4. 084961	4. 057815
H02	3. 701045	3. 697348	3. 700201	3. 711782	3. 684405
H03	3. 362282	3. 353724	3. 366501	3. 384604	3. 252341
H04	3. 043207	3. 030161	3. 051127	3. 066650	2. 797154
H05	2. 765020	2. 747015	2. 780768	2. 782728	2. 399077
H06	2. 526666	2. 503838	2. 550996	2. 539774	2. 082590
H07	2. 316560	2. 289905	2. 347560	2. 329636	1. 828741
H08	2. 129320	2. 100290	2. 164271	2. 144278	1. 622021
H09	1. 960197	1. 929887	1. 998397	1. 977245	1. 453871
H10	1. 808530	1. 777960	1. 848439	1. 826546	1. 315145
H11	1. 672759	1. 642606	1. 713537	1. 691420	1. 199349
H12	1. 551809	1. 522440	1. 592365	1. 570301	1. 101658
H13	1. 444198	1. 415917	1. 484086	1. 462134	1. 018016
H14	1. 348750	1. 321736	1. 387350	1. 365729	0. 946151
H15	1. 263872	1. 238185	1. 301051	1. 279752	0. 883548

この結果から南の引用したバーナードの計算について疑問が生じる。英語の1次エントロピーは4.124であり，今回採用したどのテキストの1次エントロピーよりも大きい。ところが2次エントロピーは今回のどのテキストの2次エントロピーよりもはるかに小さい。3次エントロピーはまあまあ近い値と言えなくはない。しかし，今回の計算では3冊を除いた残りの19冊はいずれも7次エントロピーで2.35よりも小さくなっている。どのようなテキストについてバーナードが計算したのか非常に興味がある。恐らく平均単語長が大きく，字数も2メガバイトを超えるものであろう。

堀の引用した「定説」は1.3であるが，全テキストがそれ以下になっている。

判るように一番ファイル・サイズの小さいF4がエントロピーの収束の速度が一番速い。試みにQ1のMan on Fireの第15章が約1万バイトなので，Q0として計算した結果を最後に示してあるが，15次で約0.88である。
ブロック化にして，エントロピーを計算した場合，整列対象のファイル・サイズはほとんど大きくならない。とすれば，ブロック化で計算すれば，エントロピーの収束速度は非ブロック化よりも格段に速いと予想できる。バーナードはブロック化で計算したのではないと想像できる。

情報処理教育への応用

ここまでの計算の過程で使用したアイコンは，VisualC ++とMS-DOSである。夜間あるいは旅行中の有効活用のためには，さらにバッチ・ファイルを作成し，自動運転させる必要がある。最近の「文科系」大学における教育は，あまりにもspread-sheetに偏りすぎている感がある。本報告が挑戦したエントロピー計算は，手続き型言語によらなければ不可能である。

結論

バーナードの8次エントロピー計算による2.35には，第7次エントロピーの計算で追いついたといえる。

今後の計画

ブロック化による15次エントロピーまでの計算と語の1次，2次，3次エントロピーの計算を目標にしている。

参考文献

(1) C. E. Shannon : The Mathematical Theory of Communication, Bell Systems Technical Journal, July and October (1948)

(2) 南　敏：情報理論，産業図書（1995）

(3) ヤグロム（井関・西田訳）：情報理論入門，みすず書房（1958）

(4) 堀　淳一：エントロピーとは何か，講談社（1994）

(5) 今井秀樹：情報理論，昭晃堂（平成7年）

(6) J. R. Pierce：Symbols, Signals and Noise, Harper & Row (1961)

(7) M. R. Brent：Computational Approaches to Language Acquisition, The MIT Press (1997)

(8) 北川善太郎：電子著作権管理システムとコピーマート，情報処理 Vol38, No. 8 (1997)

付録

```
0001 : ／* Creasy15. c 英語 15 連字集計 *／
0002 : # include ＜stdio. h＞
0003 : # include ＜string. h＞
0004 : # include ＜time. h＞
0005 : # include ＜math. h＞
0006 : void main (void)
0007 : {
0008 :     int i = 0, j = 0, cnt = 0, stringcnt = 0, dummycnt = 0;
0009 :     long int nowtime, start, finish;
0010 :     char chara, infile [50], work0 [50], outfile [50],
0011 :          diskette [50];
0012 :     FILE * fp0, * fp1, * fp2, * fp3;
0013 :
0014 :     printf (" Input－file name : ");
0015 :     scanf (" %s", infile);
0016 :     printf (" %s ¥n", infile);
0017 :
0018 :     printf (" Output－file name : ");
0019 :     scanf (" %s", outfile);
0020 :     printf (" %s ¥n", outfile);
0021 :
0022 :     printf (" Diskette file name : ");
0023 :     scanf (" %s", diskette);
0024 :     printf (" %s ¥n", diskette);
0025 :
0026 :     fp0 = fopen (infile," r");
0027 :     fp1 = fopen (outfile," w");
0028 :     fp2 = fopen (diskette," w");
0029 :     fp3 = fopen (" filesize. txt"," w");
0030 :
0031 :     work0 [0] = 32;
0032 :     work0 [1] = 32;
0033 :     work0 [2] = 32;
0034 :     work0 [3] = 32;
0035 :     work0 [4] = 32;
0036 :     work0 [5] = 32;
0037 :     work0 [6] = 32;
0038 :     work0 [7] = 32;
0039 :     work0 [8] = 32;
0040 :     work0 [9] = 32;
```

```
0041 :    work0 [10]  = 32 ;
0042 :    work0 [11]  = 32 ;
0043 :    work0 [12]  = 32 ;
0044 :    work0 [13]  = 32 ;
0045 :
0046 :    time (& start) ;
0047 :    if (fp0  ! = NULL)
0048 :     {
0049 :        chara = getc (fp0) ;
0050 :        while (chara  ! = EOF)
0051 :         {
0052 :           if (65 <= chara && chara <= 90)
0053 :            {
0054 :              chara = chara + 32 ;
0055 :           }
0056 :           if (chara == 0x0a)
0057 :            {
0058 :              chara = 32 ;
0059 :           }
0060:          if ((97 <= chara && chara <= 122)   | |  (chara == 32))
0061 :            {
0062 :              work0 [14]  = chara ;
0063 :              work0 [15]  = ' ¥0' ;
0064 :              fputs (work0, fp1) ;
0065 :              fputc (' ¥n', fp1) ;
0066 :              stringcnt = stringcnt + 1 ;
0067 :           }
0068 :           work0 [0]  = work0 [1] ;
0069 :           work0 [1]  = work0 [2] ;
0070 :           work0 [2]  = work0 [3] ;
0071 :           work0 [3]  = work0 [4] ;
0072 :           work0 [4]  = work0 [5] ;
0073 :           work0 [5]  = work0 [6] ;
0074 :           work0 [6]  = work0 [7] ;
0075 :           work0 [7]  = work0 [8] ;
0076 :           work0 [8]  = work0 [9] ;
0077 :           work0 [9]  = work0 [10] ;
0078 :           work0 [10]  = work0 [11] ;
0079 :           work0 [11]  = work0 [12] ;
0080 :           work0 [12]  = work0 [13] ;
0081 :           work0 [13]  = work0 [14] ;
0082 :
0083 :           chara = getc (fp0) ;
```

```
0084 :        }
0085 :        fprintf (fp3," %d ¥n", stringcnt) ;
0086 :    }
0087 :    time (& finish) ;
0088 :    time (& nowtime) ;
0089 :    fclose (fp0) ;
0090 :    fclose (fp1) ;
0091 :    printf (" (Creasy15. c)  %s infile : %s outfile : %s",
0092 :                        ctime (& nowtime), infile, outfile) ;
0093 :    fprintf (fp2," (Creasy15. c)  %s infile : %s outfile : %s",
0094 :                        ctime (& nowtime), infile, outfile) ;
0095 :    printf (" ¥nTime : %4. 0f Wordcnt = %i strings ¥n",
0096 :                      difftime (finish, start), stringcnt) ;
0097 :    fprintf (fp2," Time : %4. 0f Wordcnt = %i strings ¥n",
0098 :                       difftime (finish, start), stringcnt) ;
0099 :    fclose (fp2) ;
0100 :    fclose (fp3) ;
0101 : }
```

```
0001 : /* Pinta15. c 15次エントロピー計算*/
0002 : # include <stdio. h>
0003 : # include <string. h>
0004 : # include <time. h>
0005 : # include <math. h>
0006 :
0007 : void main (void)
0008 : {
0009 :     double entropy = 0, proba = 0;
0010 :     int totalcnt, stringcnt = 0, uniquecnt = 0;
0011 :     long int nowtime, start, finish;
0012 :
0013 :     char   infile [50], outfile [50], work0 [50],
0014 :          trace [50], diskette [50];
0015 :     const char * pinta;
0016 :     FILE * fp0, * fp1, * fp2, * fp3;
0017 :
0018 :     printf (" Input ー file name : ");
0019 :     scanf (" % s", infile);
0020 :     printf (" % s ¥ n", infile);
0021 :
0022 :     printf (" Output ー file name : ");
0023 :     scanf (" % s", outfile);
0024 :     printf (" % s ¥ n", outfile);
0025 :
0026 :     printf (" Diskette ー file name : ");
0027 :     scanf (" % s", diskette);
0028 :     printf (" % s ¥ n", diskette);
0029 :
0030 :
0031 :     fp0 = fopen (infile," r");
0032 :     fp1 = fopen (outfile," w");
0033 :     fp2 = fopen (diskette," w");
0034 :     fp3 = fopen (" filesize. txt"," r");
0035 :
0036 :     time (& start);
0037 :     if (fp0 == NULL)
0038 :      {
0039 :      {
0040 :     else
0041 :      {
0042 :        fscanf (fp3," % d", & totalcnt);
0043 :        printf (" % 7d ¥ n", totalcnt);
```

```
0044 :        pinta = fgets (trace, 50, fp0) ;
0045 :        strcpy (work0, trace) ;
0046 :        while (pinta  ! = NULL)
0047 :         {
0048 :            if (strcmp (trace, work0)  == 0)
0049 :             {
0050 :                stringcnt = stringcnt + 1 ;
0051 :            }
0052 :            else
0053 :             {
0054 :                fprintf (fp1," %7d %s", stringcnt, trace) ;
0055 :                proba =  (float) stringcnt /  (float) totalcnt ;
0056 :                entropy = entropy + proba * log (proba)  / log (2) ;
0057 :                strcpy (trace, work0) ;
0058 :                stringcnt = 1 ;
0059 :                uniquecnt = uniquecnt + 1 ;
0060 :            }
0061 :            pinta = fgets (work0, 50, fp0) ;
0062 :        }
0063 :        printf (" %7d %s", stringcnt, trace) ;
0064 :        fprintf (fp1," %7d %s", stringcnt, trace) ;
0065 :        proba =  (float) stringcnt /  (float) totalcnt ;
0066 :        entropy = entropy + proba * log (proba)  / log (2) ;
0067 :    }
0068 :    fclose (fp0) ;
0069 :    fclose (fp1) ;
0070 :    entropy = entropy / 15. 0 ;
0071 :    fp2 = fopen (diskette," w") ;
0072 :    time (& finish) ;
0073 :    time (& nowtime) ;
0074 :    printf (" (Pinta15. c)  %s", ctime (& nowtime)) ;
0075 :    fprintf (fp2," (Pinta15. c)  %s ¥n", ctime (& nowtime)) ;
0076 :   printf (" time : %4. 0f filename : %s 15th order entropy =%8.6f ¥n",
0077 :            difftime (finish, start), infile, entropy) ;
0078 :   fprintf(fp2,"time:%4. 0f filename:%s 15th order entropy =%8.6f ¥n",
0079 :            difftime (finish, start), infile, entropy) ;
0080 :    printf (" total : %d strings, unique : %d strings ¥n",
0081 :             totalcnt, uniquecnt) ;
0082 :    fprintf (fp2," total : %d strings, unique : %d strings ¥n",
0083 :             totalcnt, uniquecnt) ;
0084 :    fclose (fp2) ;
0085 : }
```