Copyright © 1978 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 57, No. 6, July-August 1978
Printed in U. S. A.

UNIX Time-Sharing System:

Statistical Text Processing

By L. E. McMAHON, L. L. CHERRY, and R. MORRIS
(Manuscript received December 5, 1977)

Several studies of the statistical properties of English text have used the UNIX system and UNIX programming tools. This paper describes several of the useful UNIX facilities for statistical studies and summarizes some studies that have been made at the character level, the character-string level, and the level of English words. The descriptions give a sample of the results obtained and constitute a short introduction, by case-study, on how to use UNIX tools for studying the statistics of English.*

I. INTRODUCTION

The UNIX system is an especially friendly environment in which to do statistical studies of English text. The file system does not impose arbitrary limits on what can be done with different kinds of files and allows tools to be written to apply to files of text, files of text statistics, etc. Pipes and filters allow small steps of processing to be combined and recombined to effect very diverse purposes, almost as English words can be recombined to express very diverse thoughts. The C language, native to the UNIX system, is especially convenient for programs which manipulate characters. Finally, an accidental but important fact is that many UNIX systems are heavily used for document preparation, thus ensuring the ready availability of text for practicing techniques and sharpening tools.

This paper gives short reports on several different statistical

* UNIX is a trademark of Bell Laboratories.

projects as examples of the way UNIX tools can be used to gather statistics describing text. A section describing briefly some of the more important tools used in all the projects is followed by three sections dealing with a variety of studies. The studies are divided according to the level of atomic unit they consider: characters, character strings, and English words. The order of sections is also in almost the chronological order of when the projects were done; future work will almost surely push forward toward more and more meaningful treatment of English.

II. TOOLS FOR GATHERING STATISTICS

2.1 Word breakout

Throughout this paper, *word* means a character string. Different words are made up of different characters or characters in a different order. For example, *man* and *men* are different words; *cat* and *cat's* are different words. We have arbitrarily taken hyphens to be word delimiters, so that *single-minded* is two words: *single* and *minded*. An apostrophe occurring within an alphabetic string is part of the word; an apostrophe before or after a word is not. Digits are discarded. Upper- and lower-case characters are considered to be identical, so that *The* and *the* are the same word. All these decisions could be made differently; the authors believe that the events are rare enough that no substantive conclusions would be changed.

The program that implements the definition of word just given is prep. It takes a file of text in ordinary form and converts it into a file containing one word per line. Throughout the rest of this paper, "word" will mean one line of a prep output file.

Optionally, prep will split out only words on a given list, or all the words not on a given list:

only option:	prep -o list
ignore option:	prep -i list

Another option which will be referred to below is the -d option, which gives the sequence number of each output word in the running input text.

2.2 Sorting

Central to almost all the examples in the rest of the paper is the sort program. sort is implemented as a filter; that is, it takes its input from the standard input, sorts it, and writes the sorted result to the standard output. The ability to send sorted output easily to a terminal, a file, or through another program is essential to make statistics-gathering convenient. The same sort program works on either letters or numbers. Among the many other features of the sort program which are used in the following are the flags:

-n:	sort a leading field numerically
-r:	sort in reverse order (largest first)
-u:	discard duplicate lines

The sorting method used is especially well adapted to the kind of files dealt with in statistical investigations of text. Its skeleton, which decides which elements to compare, takes advantage of repetition of values in the file to be sorted. The algorithm used for in-core sorting is a version of Quicksort which has been modified to run faster when values in the input are repeated. The standard version of Quicksort requires $n \log n$ comparisons, where n is the number of input items; the UNIX version requires at most $n \log m$ comparisons, where m is the number of *distinct* input values.

2.3 Counting

Another tool of interest for many statistics-gathering processes is a program named uniq. Its fundamental action is to take a sorted file and produce an output containing exactly one instance of each *different* line in the file. (This process simply duplicates the action of sort with the -u option; it runs much more quickly if a sorted file is already available.) More often useful is its ability to count and report the number of occurrences of each of the output lines (uniq -c).

A very generally useful tool is the program wc. It simply counts the number of lines, words, and characters in a file. Throughout any investigation of text statistics, the question arises again and again: How many? Either as a command itself or as the last filter in a chain of pipes, wc is invaluable for answering these questions.

2.4 Searching and pattern-matching

A program of common use for several purposes is grep. grep searches through a file or files for the occurrence of strings of characters which match a pattern. (The patterns are essentially the same as the editor's patterns and, indeed, the etymology of the name is from the editor command **g/r.e./p** where **r.e.** stands for regular expression.) It will print out all matching lines, or, optionally, a count of all matching lines. For example,

```
prep document | grep "^....$" | sort | uniq −c >fours
```

will find all of the four-letter words in **document** and create a file named **fours** which contains each such different word along with its frequency of occurrence.

sed, the stream editor, is a program which will not only search for patterns (like **grep**), but also modify the line before writing it out. So, for example, the following command (using the file **fours** created by the previous example) will print only the four-letter words which appear exactly once in the document (without the frequency count):

```
sed −n "s/^ *1 //p" fours
```

This ability to search for a given pattern, but to write out the selected information in a different format (e. g., without including the search key), makes **sed** a useful adhesive to glue together programs which make slightly different assumptions about the format of input and output files.

III. CHARACTER LEVEL

Frequency statistics of English text at the character level have proved useful in the areas of text compression and typographical error correction.

3.1 Compression

Techniques for text compression capitalize on statistical regularity of the text to be compressed, or rather its predictability. The statistical text-processing programs on UNIX have found use in the design and implementation of text-compression routines for a variety of applications.

Suppose that a file of text has been properly formatted so that it does not contain unnecessary leading zeros and trailing blanks and the like, and that it does not devote fixed-length fields to variable-length quantities. Then the most elementary observation that leads to reducing the size of the file is that the possible characters of the character set do not all occur with equal frequency in the text. Most text uses ASCII or other 8-bit representation for its characters and typically one of these eight bits is never used at all, but one can go much further. If we take as a measure of the information content of a string of characters

$$H=\sum -p_i \log_2 p_i$$

where p_i is the probability of occurrence of the character x_i and the sum is taken over the whole character set, then it is theoretically possible to recode the text so that it requires only H bits per character for its representation. It is possible to find practical methods which come close to but do not attain the value of H. Of course, in deriving this estimate of information content, we have ignored any regularity of the text which extends over more than one character, like digram statistics or words.

It is a simple matter to compute the value of H for any file whether it is a text file or not. The value of H turns out to be very nearly equal to 4.5 for ordinary technical or non-technical English text. This leads immediately to the possibility of recoding the text from ASCII to a variable-length encoding so as to approach a compression to 56 percent of the original length.

Data files other than English text usually have quite different statistics from English text. For example, telephone service orders, parts lists, and telephone directories all have character statistics which are quite different from those of English and different from each other. In general, data files have values of H smaller than 4.5; when they contain a great deal of numerical information, the values of H are often less than 4.

Programs have been written on UNIX to count the occurrences of single letters, digrams and trigrams in text. Single-letter frequencies are kept for all 128 possible ASCII characters. For the digram and trigram statistics, only the 26 letters, the blank, and the newline characters are used, and upper-case letters are mapped to lower case.

The result of running this program on a rather large body of text is shown in Table I. The input was nine separate documents with a total of 213,553 characters and 36,237 words. The documents consisted of three of the Federalist Papers, each by a different author, an article from this journal, a technical paper, a sample from Mark Twain, and three samples of graded text on different topics.

Some interesting (but not novel) observations about the nature of English text can be made from these results. At the single-character level, some characters appear in text far more often than others. In fact, the 10 most frequent characters constitute 70.6 percent of the

Table I—English text statistics

Sample character, digram, and trigram counts for a sample of English text. The counts are truncated after the first 25 entries. 012 is the newline character. □ in the character column is a space character; in the digram and trigram columns, it is any word separation character.

count	character	cum. %	count	digram	count	trigram
33310	□	15.5	6156	e□	3661	□th
21590	e	25.7	5364	□t	3617	the
16080	t	33.2	4998	th	2504	he□
13260	a	39.4	4099	he	1416	□of
12584	o	45.3	3801	□a	1353	of□
12347	n	51.1	3748	s□	1301	□in
12200	i	56.8	3367	in	1249	and
10997	s	61.9	2780	er	1225	□an
10640	r	66.9	2757	t□	1144	nd□
7930	h	70.6	2738	d□	1088	□to
6622	l	73.7	2708	re	1027	to□
5929	d	76.5	2666	an	1025	ion
5409	c	79.0	2572	□i	1003	ed□
4524	012	81.2	2517	n□	946	ing
4508	u	83.3	2506	□o	875	ent
4152	m	85.2	2244	on	854	is□
4080	f	87.1	2047	es	851	in□
3649	p	88.8	2025	at	830	tio
3090	g	90.3	1990	en	805	□co
2851	y	91.6	1912	□s	779	re□
2654	w	92.9	1840	y□	747	□a□
2483	b	94.0	1835	ti	734	ng□
1984	,	94.9	1799	nd	709	on□
1884	v	95.8	1723	nt	702	□be
1824	.	96.7	1681	te	701	es□

text and the 20 most frequent characters make up 91.6 percent of the text. At the digram level, of the 784 possible 2-letter combinations, only 70 percent actually occur in the text. More dramatically, at the trigram level, of the 21952 possible combinations, only 4923, or 22.4 percent, occur in the text. One implication is that, instead of the 24 bits used to represent a trigram with an 8-bit character set, a scheme using 13 bits would do, a compression to 54 percent of the original length, using only the fact that less than 2^{13} different trigrams occur in the text. Noting the widely varying frequencies of the trigrams in the text, we can obtain a considerably better compression rate by using a variable-length encoding scheme.

3.2 Spelling error detection

The observation that English text largely consists of a relatively small proportion of the possible trigrams led to the development of a program typo which is used to find typographical errors in documents. A single erroneous keystroke on a typewriter, for example, changes the three trigrams in which it occurs; more often than not, at least one of the erroneous trigrams will be otherwise extremely rare or nonexistent in English text. The same thing happens in the case of an erroneously omitted or repeated letter. Better performance is obtained when the comparison statistics are taken from the document itself rather than using some set of trigram statistics from English text in general.

typo accumulates the digram and trigram frequencies for a document and uses them to calculate an index of peculiarity for each word in the document.[1] This index reflects the likelihood that the trigrams in the word are from the same source as the trigrams in the document. Words with rare trigrams tend to have higher indexes and are at the top of the list.

On systems large enough and rich enough to keep a large English dictionary on line, the same function of finding likely candidates for spelling correction is performed by a nonstatistical program, spell, which looks up every word in the dictionary. Of course, suffixes like *-ing* and *-ed* must be recognized and properly stripped before the lookup can be done. What is more, very large dictionaries perform poorly because so many misspelled words turn out to be names of Chinese coins or obsolete Russian units of distance. Not surprisingly, the statistically based typo requires little storage and runs considerably faster. Moreover, not all systems have such resources, and typo has proven useful for authors and secretaries in proofreading. A sample of output from the typo program is included as Table II.

IV. STATISTICS OF CHARACTER STRINGS

In this section we consider statistics which take character strings as atomic units, without any reference to the string's use or function as an English word.

4.1 Word-frequency counts

A set of statistics from a text that is frequently collected (often as a base for further work) is a word-frequency count. A list is made of all the different words in the text, together with the number of times each occurs.[2] With the UNIX tools, it is quite convenient to make such a count:

```
prep text−files | sort | uniq −c
```

This command line produces a frequency count sorted in

Table II—Typo output

A portion of the output of the typo program from a 108-page technical document. A total of 30 misspelled words were found, of which 23 occurred in this portion. The misspelled words identified by the author of the document upon scanning the list have been marked by hand.

```
Apr 12 22:32:11  Possible typo's and spelling errors Page 1

☛ 17 nd                    14 flexible          ☛ 5 pesudonym
  17 heretofore            14 flags             ☛ 5 neames
☛ 17 erroronously          14 conceptually      ☛ 5 namees
☛ 16 suer                ☛ 14 bwaite              5 multiplied
  16 seized                14 broadly             5 interrelationship
☛ 16 poiter              ☛ 14 amy                 5 inefficient
  16 lengthy               14 adds                5 icalc
  16 inaccessible          14 accompanying        5 handler
  16 disagreement          13 overwritten         5 flag
☛ 16 bwirte                13 occupying           5 exercised
  15 violating             13 lookup            ☛ 5 erroreous
  15 unaffected            13 flagged             5 dumped
  15 tape                ☛ 9 iin                  5 dump
  15 swapped             ☛ 8 subrouutine          5 deficiency
  15 shortly               8 adjunct              5 controller
☛ 15 mutiliated            7 drawbacks            5 contiguous
  15 multiprogramming    ☛ 6 thee                 5 changing
  15 likewise            ☛ 6 odification          5 bottoms
  15 datum               ☛ 6 od                 ☛ 5 bitis
☛ 15 dapt                  6 indicator            5 ascertain
  15 cumulatively          6 imminent           ☛ 5 accomodate
  15 consulted             6 formats              4 unnecessarily
  15 consolidation         6 cetera               4 traversing
  15 checking              5 zeros                4 tracing
☛ 15 accordinng            5 virtually            4 totally
☛ 14 typpical              5 ultimately           4 tops
  14 tabular               5 truncate             4 thirteen
  14 supplying             5 therewith          ☛ 4 tallyed
  14 subtle                5 thereafter           4 summarized
  14 shortcoming           5 spectre              4 strictly
  14 pivotal               5 rewritten            4 simultaneous
  14 invalid               5 raises               4 retrieval
  14 infrequently          5 prefix               4 quotient
```

alphabetical order, as in Table IIIa. To obtain the count in numerical order (largest first):

```
prep text−files | sort | uniq −c | sort −n −r
```

This is illustrated in Table IIIb.

4.2 Dictionary compression

A more complex but considerably more profitable approach to text compression is based on word frequencies. Text consists in large part of words; these words are easy to find in the text; the total number of different words in a text is several orders of magnitude

Table III—Word frequency counts

The beginning of (a) alphabetically sorted and (b) numerically sorted word frequency counts for an early draft of this paper.

(a)		(b)	
124	a	321	the
3	ability	212	of
3	about	124	a
3	above	114	in
1	abrupt	105	to
1	abstract	80	is
1	accidental	78	and
1	according	65	words
1	accordingly	63	text
1	account	50	for

less than the total possible number of arbitrary character strings of the same length. The approach can best be visualized by supposing that a file of text consists entirely of a sequence of English words. Then we can look up each word in a dictionary and replace each word in the text by the serial number of the word in the dictionary. Since a dictionary of reasonable size contains only about 2^{16} words, we have found an immediate and trivial method to recode English text so as to occupy 16 bits per word. Since the average length of a word in text, including the blank after it, is 6 characters, we have a representation that requires only about 2.7 bits per character. This implies a compression to 37 percent of original length. Some details, of course, could not be neglected in actual practice, like capitalization, punctuation, and the occurrence of names, abbreviations, and the like. It turns out, however, that these are sufficiently rare in ordinary running text that only about two or three extra bits per word are required, on the average, to handle them and it is possible to attain a representation requiring only about 3 bits per original character.

In the case of technical text, it is profitable to find the words from the text itself, and store them, each word once, in the compressed file. When this is done, the total number of different words is rather small and because of the tendency of technical authors to use a small technical vocabulary very heavily, the performance is very good. If the dictionary is stored in the file, then the compression performance depends on the number of times each word is used in the text. Suppose there is a word in the text which is m characters long and occurs n times. Then, the occurrences of that word occupy $m \times n$ characters in the original text, whereas in the compressed text, m characters are used for the one dictionary entry and $n \times k$ bits are used as a dictionary pointer each time the word occurs in the text, where k is the logarithm (base 2) of the number of dictionary entries.

Of course, words in a text do not occur with equal frequency and it is possible, just as was done with letter statistics, to use a variable-length encoding scheme for the words. The information content of the words in a text can be found by passing the word-frequency count found in the previous section through one more filter:

```
prep file−name | sort | uniq −c | entropy
```

It turns out that, for nontechnical English text, the information content of the words is between 8 and 9 bits per word when it is estimated from the text itself. This implies that a text consisting entirely of a string of English words can generally be compressed to occupy only about 1.5 bits per original character. Needless to say, the amount of processing required to compress and expand text in such a way is usually prohibitively high.

4.3 Specialized vocabulary

A practical application of word-frequency counts arose when colleagues became interested in devising vocabulary tests for Bell System personnel to determine their familiarity with the vocabulary used in Bell System Practices (BSPs) in various areas. It is intuitively clear that the vocabulary used in Bell System Practices differs from the general English vocabulary in several details. Some words, like *the*, *of*, *an*, etc., are common in the language in general and in specialized writing; others, like *democracy, love, mother* would be found much more frequently in the language in general than in BSPs; others, like *line, circuit, TTY* would be more frequent in BSPs than in the language generally. What was desired was an automatic procedure which would identify such words without relying on intuition. The general problem proposed was to identify the specialized vocabulary of a specific field; the immediate interest was in words with moderate frequencies in BSPs dealing with a certain area, and which are much less frequent in the language as a whole. It was hoped that familiarity with such words would indicate familiarity with the field.

A word-frequency count of approximately one million words of English text was available. It was made from the text of the Brown Corpus[3] and closely resembles the published frequency count of that corpus. It differs in detail only because we used prep's definition of

Table IV—Indexes (see text) of specialization

Frequency of words in a half-million words of BSPs, frequency in a million words of general English, and the words for (a) words which occur too often in BSPs relative to general English; and (b) words which appear too seldom.

(a) Index	(a) BSP Frequency	(a) English Frequency	(a) Word	(b) Index	(b) BSP Frequency	(b) English Frequency	(b) Word
4362	2585	73	fig	-1005	218	2670	their
4150	2334	8	trunk	-1008	23	1909	what
3933	2643	233	control	-1034	584	3941	have
3541	2216	120	test	-1085	4	1961	said
3399	1950	25	circuit	-1207	132	2719	would
3277	2326	266	b	-1212	18	2252	who
3106	2059	168	equipment	-1234	14439	36472	of
3094	1881	76	frame	-1356	298	3617	they
2990	1684	7	cable	-1395	34	2652	we
2828	1785	104	unit	-1457	2	2619	him
2472	1940	315	line	-1593	1	2858	she
2445	1367	1	ess	-1658	71	3284	were
2418	1479	64	message	-1696	0	3037	her
2213	10707	10098	is	-1716	2389	10596	that
2190	1316	46	wire	-1788	287	4381	but
2133	1892	418	system	-1809	10	3285	you
2129	1443	133	list	-2096	1439	8768	it
2117	1513	178	data	-2502	177	5247	i
2018	2085	612	used	-2797	27	5132	had
1936	1118	18	lamp	-3839	27	6999	his

a word, which is slightly different from that of Kučera and Francis. This frequency count was used as representative of English as a whole.

Also available were approximately a half-million words of BSPs, from plant, station, and ESS (Electronic Switching System) maintenance areas. Frequency counts were made of these three areas separately and of the BSP text as a whole.

An index of peculiarity was defined for each word as follows: The two frequency distributions were considered as a single two-way classification (source by word), and single-degree-of-freedom χ^2 statistics were computed for each word. To distinguish words that appear too frequently in the BSPs from words that appear too seldom, a minus sign was attached to the index when the word appeared less often than might be expected in the BSPs (with reference to the English frequency count). This index has the advantage of automatically taking into account differences in size of the two frequency counts, and also de-emphasizing moderate differences in frequency of words which occur rarely in either set of texts.

Samples of the output are shown in Table IV. The indexes and frequencies are for the entire half million words of BSPs compared with English. The first word in the table, "fig," does not mean that the BSPs discussed tropical fruit to any extent; it is a sample of the difficulties of defining words as character strings. The word is prep's version of Fig. (as in "Fig. 22"). The next several words are, as expected, nouns which refer to objects prominent in telephony. The occurrence of *is* more frequently in BSPs than would be expected seems to be a comment on the style of the writing, one with many passive constructions and predicate noun or adjective constructions—a quite abstract style.

The general method for comparing the vocabulary of specialized texts with general English text worked well for the specific problem proposed; it allowed our colleagues to choose words for their test conveniently. It also shows promise as a more generally applicable method.

4.4 Readability

The number of *tokens* (N) in a text is the number of running words; the number of *types* (T) is the number of different words. For example, the sentence

The man bit the dog.

contains five tokens, and only four types: *the*, *man*, *bit*, *dog*. For our purposes, the number of tokens in a file is taken to be the number of words in the output file of the prep command. We will call a type which occurs exactly once in a text a *hapax,* from the Greek *hapax legomenon,* meaning occurring only once. The number of hapaxes in a text is H.

Two summary statistics often calculated from word-frequency counts are the type/token ratio T/N and the hapax/type ratio H/T. The T/N ratio gives the average repetition rate for words in a text; its usefulness is limited by its erratic decrease with increasing N. The H/T ratio also varies with N, but more slowly and less erratically. We have found it to be of interest in investigations of readability.

Readability is an index which is calculated from various statistics of a text, and is intended to vary inversely with the difficulty of the text for reading. Several such indexes have been proposed; none is universally satisfactory (see, for example, Ref. 4).

In the following discussion, the text used to measure the effectiveness of proposed indexes of readability is taken from an extensive study by Bormuth.[4] He gathered passages from published works in several different fields, which were intended by the publisher to be appropriate for reading by students at various grade levels. He carefully graded the difficulty of each text by an independent psychological criterion and calculated an index of difficulty from the results of the psychological tests. To judge the effectiveness of indexes calculated from the statistics of the texts themselves, we used two criteria: Bormuth's psychological index and the publishers' assignment of the text to grade level. (The publishers' assignment is in general not based on empirical tests, but on considerable experience and art. It correlates well with Bormuth's empirical measures; we use it simply as a check on oddities that might arise from the specific nature of Bormuth's index.)

One factor which, intuitively, makes some text more difficult to read than other is the speed with which new ideas are introduced. A passage which deals with several ideas in a few words tends to be more difficult to comprehend than a passage of the same length which spends more time developing a single idea. The computer, which of course does not understand the text, cannot measure the number of ideas in a passage. But several statistics regarding the number of different words used, and the number of times they are repeated, might plausibly be expected to vary with the number of ideas in a passage.

Of the statistics related to the breadth of vocabulary in a passage, the H/T ratio was found to correlate best with Bormuth's empirical measure of readability. Over twenty 275-word passages, the correlation is -0.79 with Bormuth's index, 0.77 with grade placement. This correlation is high for a single statistic with an empirical measure of readability, and the correlation remains whether or not the rationale given above is convincing for why there should be a correlation.

We return to readability in the last section of this paper.

4.5 Dispersion of words

A final example of statistics based on words purely as character strings concerns the dispersion of words in text. When an author writes a passage, it is plausible to believe that he has an over-all topic, which unites the passage as a whole. As he proceeds, however, he attends to first one aspect of the topic, then another. It might be expected that as he concentrates on one aspect he would use words from one part of his vocabulary, and when he shifts to another aspect, the vocabulary would also change somewhat. (See Ref. 5, pp. 22-35.) This tendency might be measured by observing

Table V—Separation between words

Mean and standard deviation for the separation (as fractions of the document) between words that occurred exactly twice in documents. *N* is the number of words on which the mean and S.D. are based. The 275 word entries are averages for 4 passages from different sources; the mixed entries are for a concatenation of the four passages; matched entries are for a continuous 1,200-word passage; expected entries are on the hypothesis of random placement of words.

	N	Mean	S. D.
275 word	21	0.26	0.20
mixed	62	0.15	0.16
matched	62	0.32	0.23
expected		0.33	0.24

the distance between repeated occurrences of words. If the tendency to change vocabulary is strong, repeated instances of the same word would be closer together than when the topic and therefore the vocabulary is uniform over an entire passage. In any case, since an English text is presumably an organized sequence of words, the dispersion of words should be less than would be expected for a random placement of the words in a passage.

To gather statistics on the dispersion of words, an option of **prep** will write the sequence number of each word (in the input stream) on the same line as the word. By using this option together with the **-o (only)** option, the position in the input text of each occurrence of each word that appears twice, three times, etc. can be written into a file. This file, sorted on the word field, provides input to a simple special-purpose program to calculate the distances between repeated occurrences of the same word. The entire process required writing only one very simple special-purpose program to find the differences between sets of numbers in a file.

Sample results to illustrate the behavior of the statistic are displayed in Table V. The observed and expected means and standard deviations for the separation of words that occurred exactly twice in a text are given as fractions of the length of the text. (The expected fractions are calculated on the hypothesis of random placement of the words in the text.) The line labeled *275 word* gives the average statistics for four passages of about 275 words each, drawn from different biology texts, each on a separate topic. The *mixed* line is statistics from the concatenation of the four texts; the *matched* line gives statistics from a text of the same length as the concatenation, but drawn from a continuous, coherent text. The *expected* line gives the expected separations on the hypothesis of random placement of the words in the text.

As can be seen in Table V, the mean dispersion behaves as expected; it is smaller than random placement for all texts, but larger for coherent texts than for samples constructed to have abrupt changes of topic. The large standard deviation relative to the mean makes it a difficult statistic to work with, but it shows promise as a measure of uniformity of topic in a passage.

V. ENGLISH WORDS

In this section, we go a short step beyond the character-string orientation of the last section and consider different functional uses of our character strings as English words. Words will still be defined by **prep** as character strings separated by space or punctuation, but we attend to the way these character strings are used as English words.

5.1 Readability revisited

In the previous section, we considered the correlation of a statistic based on breadth of vocabulary with readability. Another way in which English text can become difficult to read is for an author to use long, complicated constructions which require the reader to follow tortuously through the maze of what is a single sentence, and, therefore, presumably a single thought. (The preceding sentence is offered as a rather modest example of its own referent.) English sentences become long and complicated usually by use of connective words like prepositions (of, from, to, etc.) and conjunctions (and, when, if, although, etc.). Therefore, a list was drawn up of connective words (prepositions and conjunctions), and another list of other function words (auxiliary verbs, articles, demonstratives, etc.). Using **prep -o** through **wc**, the number of connectives and the number of other function words in each of twenty graded passages[4] were counted. As expected, reading difficulty as measured both by Bormuth's psychological index and by publishers' grade placement was correlated with both indexes. Number of connectives per token was correlated -0.72 with Bormuth's index score; 0.69 with grade placement. Number of other function words per token was correlated 0.57 with Bormuth's index; -0.50 with grade placement.

The two best predictors considered, H/T ratio and density of connective words, are, alas, highly correlated with each other, so that the multiple correlation of the two predictors is only 0.80 with Bormuth's index and 0.78 with grade placement. This finding that predictors of readability are highly correlated among themselves is common.[4] It is probably unavoidable when the passages used to validate a readability index are taken from text intended for different audiences. An author, knowing that his audience is less skilled in reading, makes his text simpler in every respect: vocabulary, sentence structure, etc. At present, however, no reliably graded text is available to the authors which is intended for our target audience.

5.2 Word class assignment in English text

Several of the statistics described in Section II were collected, *faute de mieux,* on all words in the text. The work on vocabulary differences, for instance, would have profited from being done only on the nouns in the text, since the differences in concepts between specialized fields should show up especially in the nouns used to refer to the concepts. When the work described in Section IV was done, there was no automatic way to classify the words in a text as nouns, verbs, adjectives, etc.

Recently, a set of programs has been written to automatically assign word classes (parts of speech) to words. The system operates on the principle that, if word classes can be assigned or partially assigned to many of the words in a sentence, the word classes of the remaining words can be deduced. There are 38 internal word classes that are collapsed to 14 on final output. Many of the internal classes are combinations of the 14 final classes with the last program making the final decision as to what class to assign. Examples of the internal classes are plural noun-verb, singular noun-verb, noun-adjective, ed, ing. The system consists of three programs written in Lex and C.

The first program looks each word up in a dictionary of 210 functional words and 140 irregular verbs. This phase assigns classes to the common articles, pronouns, auxiliary verbs, etc.

The second program uses word endings in an attempt to assign word classes. Each word not assigned a word class by the first program is compared with 49 word endings. If a match is found, the word is checked with a list of words that are exceptions to the ending rule and a class assignment is made accordingly. (The list of exceptions was made automatically using an on-line dictionary containing part-of-speech information.) After phase 2, 55 percent of the words have unique class assignments, 29 percent have partial assignments, and 16 percent are still totally unassigned.

The third program does most of the work. Basically, it scans a sentence looking for a verb. Having found either a verb or an indication to stop scanning, it returns to the beginning of the sentence and starts assigning word classes, looking for a subject. The program assumes sentences to be either declarative or interrogative; it has no way of detecting an imperative sentence.

A sample sentence is presented below. The first line is the sentence, the second the results after the dictionary and endings phases are complete, and the last line the final output.

```
The  diffusion of   formal political controls in   the
art  noun      prep unk    adj       pl-n-v   prep art
art  noun      prep adj    noun      noun     prep art

national government serves to   magnify the influence of
noun-adj noun       pl-n-v to   verb    art n-v       prep
noun     noun       verb   verb verb    art noun      prep

organized private groups in   public   affairs .
ed        unk     pl-n-v prep noun-adj pl-n-v
adj       noun    noun   prep adj      noun
```

The system was run on several samples of graded English text containing a total of 11,705 words. Word class assignments were assigned independently by hand and the program results were compared to those of the human coder. A total of 588 errors were made by the program, a 5 percent error rate. Of these errors, 34 percent were mistaken verbs, both false positives and missed cases. The greatest number of errors were noun-adjective confusions.

Even though the program is not perfect, it is very good in matching human-assigned classes, and should be useful in extending the statistical projects described above.

VI. CONCLUSION

This paper has illustrated a variety of the uses to which the UNIX system and UNIX tools have been put in describing English text statistically. The basic concepts of UNIX programming, like standard input and output, pipes, filters, etc., which make all programming convenient, are especially useful for collecting statistics, where a small number of operations (like counting) are applied to a wide variety of inputs. The C language is also very convenient for text, because of its good character manipulation facilities. Finally, the fact that UNIX is so much used for document preparation has ensured that a large body of text is always at hand for practicing and sharpening tools.

REFERENCES

1. R. Morris and L. L. Cherry, "Computer Detection of Typographical Errors," IEEE Trans. on Professional Communication, *PC-18* (March 1975), pp. 54-56.
2. G. U. Yule, *The Statistical Study of Literary Vocabulary,* Cambridge: The University Press, 1944.
3. H. Kucera and W. Francis, *Computational Analysis of Present-Day American English,* Providence: Brown Univ. Press, 1967.
4. J. C. Bormuth, "Development of Readability Analyses," U. S. Dept. HEW Final Report, Project 7-0052, Contract EC-3-7-070052 0325, Chicago University Press (1969).
5. F. Mosteller and D. L. Wallace, *Inference and Disputed Authorship: The Federalist,* Reading, Mass.: Addison-Wesley, 1964.