A Telecommunications Aid for the Deaf:

An Algorithm For Extracting Readable English Text From Touch-Tone® Keystroke Sequences

by

Scott L. Minneman

Submitted to the Department of
Mechanical Engineering
in Partial Fulfillment of the Requirements
for the Degrees of

Bachelor of Science

and

Master of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology
November, 1984

Signature of Author .....
Department of Mechanical Engineering
November, 1984

Certified by .....
Michael James Rosen
Thesis Supervisor

Accepted by .....
Professor Ain Ants Sonin, Chairman
Departmental Committee on Graduate Studies
Department of Mechanical Engineering

A Telecommunications Aid for the Deaf:

An Algorithm For Extracting Readable English Text From Touch-Tone® Keystroke Sequences

by

Scott L. Minneman

Submitted to the Department of Mechanical Engineering
on November 20, 1984, in partial fulfillment of the requirements
for the degrees of Bachelor of Science and Master of Science in
Mechanical Engineering

Abstract

Telecommunication aids for the hearing-impaired have disadvantages that preclude full use of the telephone network by these individuals. The teletypewriter aids currently available to the deaf require that a non-impaired communicative partner have expensive and inconvenient technical support in order to communicate. Using the Touch-Tone telephone keypad for text transmission has been proposed to alleviate the situation. Touch-Tone text communication devices must somehow address the problem of distinguishing between the three letters appearing on each key. Many of the proposed systems have required additional keystrokes by the sender to signify which letter on a particular key was desired.

An algorithm was developed to relieve the sender workload imposed by multiple-keystroke letter codes in Touch-Tone communication systems. A sender using a system incorporating this algorithm simply "types" the message on the Touch-Tone keypad of a standard telephone; no encoding is required to signify which particular letter was intended. The "hybrid" algorithm, running on a microcomputer at the receiving end, extracts a readable English text string from the transmitted sequence of keystrokes using dictionary lookup and probability methods operating in parallel. This text string is displayed for the hearing-impaired user to read.

An experiment was undertaken to compare the rates of text entry using the multiple-keystroke and single-keystroke selection methods. Results showed that the single-stroke method of entering messages permitted by the algorithm is twice as fast as competitive methods. Subjective evaluations indicate lower sender workload and lower error rates.

The algorithm, implemented on a microcomputer, correctly deciphers > 95% of the words in free running speech, and produces English-like approximations for the remainder. Informal conversations simulating device use indicate that text produced by the algorithm is easily readable.

Michael James Rosen, Ph.D.
Principal Research Scientist and Lecturer
Department of Mechanical Engineering

Acknowledgements

I would like to use this opportunity to thank the following people for their contribution to this thesis:

Mike Rosen – for his tolerance and his excellent advising

Cheryl Goodenough-Trepagnier – for her helpful comments

Pat Demasco – for his programming help, his company when working late, and for numerous rides on Gretchen

Susan Klein – for her undying support, her help in turning my garble into readable text, and for being there

Ken Sinclair – very, very much – for his crucial role in producing this document

Daniel Sonnenfeld – for showing me how things are for the hearing-impaired

Table of Contents

List of Figures

/

List of Tables

1. Introduction

Telecommunications for the deaf and hearing-impaired has been a topic of interest since the invention of the telephone. Articles describing technical aids for phone use were published as early as 1963 [Smith]. High-gain telephone amplifiers and acoustic couplers for hearing aids have been commercially available for some time. Though many approaches to telecommunications for the severely hearing impaired were found in the literature [Bellefleur,1976; Bozzuto,1981; Glaser,1981; Levitt,1970; Smith,1976; Wallingford,1974], few have been developed beyond the prototype stage. The deaf community has seemingly been hampered by an antiquated communication protocol that, while initially adopted for simplicity and low cost, has resulted in inordinately expensive equipment that is incompatible with mass market data communications equipment. Now, however, with the availability and proliferation of personal computers, a breakthrough is possible in the area of telecommunications network access by this population.

1.1 The Hearing Impaired Population

The hearing impaired population of the United States numbers thirty-two million. Of this number, nine million are impaired to the extent that they experience difficulty in routine communication. Four million of those are classified "severely hearing impaired" meaning that while they are capable of distinguishing various sounds when using an aid, they need lip reading and other techniques to augment their hearing. Another two million are classified as totally deaf [Bowe,1984].

Deaf individuals are often categorized in ways other than by the severity of their impairment. Individuals are grouped with respect to the period of onset of their disability, the physiological cause of the impairment, and their preferred method of communication. These categorizations are used primarily in descriptions of the population and for determining appropriate intervention strategies for rehabilitation.

A particularly useful distinction in the context of telecommunications for the deaf concerns the hearing impaired person's level of speaking ability. If the deaf person is capable of producing utterances that are comprehensible by the general population, that person is referred to as oral deaf. If, on the other hand, the deaf person is incapable of comprehensible speech, they will be classified as non-oral deaf.

Another useful observation about the deaf population concerns their level of written language competence. The deaf population has a statistically higher incidence of all types of language impairments, including written. A communication device that requires a level of written language competency must take into account this characteristic of the population.

That portion of the hearing impaired population for which a telecommunication aid would be beneficial is estimated at over 8 million. This figure assumes that half of the totally deaf have sufficient language skills to operate an aid, that the entire "severely hearing impaired" population would benefit, and that another 3 million of the remainder of the hearing-impaired population are sufficiently impaired to benefit from a telecommunication aid.

1.2 Ramifications of Telecommunication Impairment

The telephone is an important tool for information exchange in modern society, especially in business and industry. The deaf are precluded from many employment opportunities because they are incapable of using the telephone in the conventional manner. Social conventions that developed with telephone use further isolate the deaf. The deaf community has various methods of internal telecommunication, but it remains largely isolated from the non-impaired.

The deaf are financially penalized by their impairment. The median income of deaf people is 72% of that of the non-impaired; for the prelingually deaf (generally non-oral), this figure drops to 62% [Anon,1983]. Although inability to access the telecommunications network is only partially responsible for this situation, its effect is significant.

1.3 NINCDS Advisory Report

In 1979, the National Institute of Neurological and Communicative Disorders and Stroke (NINCDS) commissioned the preparation of a planning document which addressed the present status and future direction of research with respect to sensory aids [Levitt, *et al* 1979]. Quoting from the current status section of that document about aids for the hearing impaired:

> *The development of the telephone and the development of sensory aids for the hearing-impaired have a closely intertwined history. Alexander Graham Bell was deeply concerned with the development of a speech-training aid for the deaf, and several key steps in the invention of the telephone stem from his concurrent work on sensory aids. The electronic hearing aid, the audiometer, methods for assessing speech reception skills, speech-spectrum displays, teletypewriters for the deaf – all stem from developments in telephone engineering.*
>
> *The key telecommunication aids are teletypewriters for the deaf and television. There are also a number of sensory aids that may be used directly with the telephone, such as high-gain telephone receivers and conventional hearing aids that are coupled magnetically or acoustically to the telephone receiver. Special-purpose communication aids are also available, such as Morse-code signaling systems and electronic speech synthesizers.*
>
> *Recent developments in the telecommunications field reflect a more widespead use of available technology rather than the introduction of new types of aids or techniques. In particular, there has been a rapid growth in the use of teletypewriters by the deaf. This growth has been facilitated in part by a gradual reduction in the cost of these devices, resulting from increased demand and improved mass production. However, costs are still prohibitive for many deaf individuals, and further significant reductions in both purchase price and telephone maintenance costs would greatly increase the use of these devices and their attendant benefits.* [Ward *et al* 1979]

While there may have been a "rapid growth in the use of teletypewriters for the deaf" the number of teletypewriters currently in use is estimated at less than 100,000. This figure is very small when compared to the number of individuals who would benefit from an aid. In addition, it is important to note that this figure encompasses the entire population served by the teletypewriter network, any able-bodied individual, civil group, or public service who has purchased an aid to converse with the disabled is included in this number.

The NINCDS report states in the future directions section:

> *It would be of considerable benefit to the hearing impaired populaiion if access to the telephone system could be facilitated. The continuing growth of the teletypewriter network of the deaf is a positive step in this direction, but costs to the user are high. The bandwidth requirements for a teletypewriter are much less than those for a conventional speech channel, and methods for reducing costs by more efficient use of the telephone system by deaf users should be investigated.*
>
> *An inherent limitation of the teletypewriter network is that special equipment is needed to access the network, and the deaf teletypewriter user is essentially cut off from other users of the telephone system unless a central translating system is made available. Methods for using conventional telephones, such as the pushbutton telephone, to access the deaf teletypewriter network should be developed.* [Ward *et al* 1979]

It is important that the current teletypewriter network not be considered so firmly established that other possibilities for telephone use for the deaf be discounted. This attitude is expressed to some extent in the final sentence of the previous quote. There are not so many teletypewriters in use today that future developments in telecommunications for this population need to be dictated by their presence.

1.4 Conclusions

There is a sizeable population that will benefit from improvements in telecommunication aids for the deaf. The influence of hearing impairment on the social and occupational lives of this population are significant and debilitating. Although research is progressing on choclear implants and other techniques to relieve hearing impairment at the physiological level, the need for telecommunication aids will remain for the forseeable future.

2. Background

The deaf and hearing-impaired currently have numerous techniques for telecommunication at their disposal, and additional methods are under development. An examination of these is necessary to identify problem areas, both those that need attention and those where a solution is forthcoming, in which additional effort would be wasted.

2.1 History of Telecommunications for the Deaf

In this section, the hearing-impaired individual will frequently be referred to as the receiver, and a non-impaired individual will be referred to interchangably as the sender or caller. This is not intended to relegate the communicative roles of either, but is done rather as a matter of convenience.

2.1.1 Interpreter

A very common, and probably the earliest, method of phone use by the deaf employs a non-impaired person to relay the incoming phone message. This human interpreter listens to the message, then repeats it to a deaf person capable of lip reading or signs it to a manually deaf receiver. The deaf person responds orally to the caller or the interpreter verbalizes a response that the deaf receiver has written or signed.

This technique, though simple, reliable, and inexpensive, has several obvious disadvantages. The deaf individual is dependent upon an interpreter being available whenever a call is received or is to be made. The author, having lived in company with a deaf person using solely this technique, has observed potential interpreters actively avoiding participating in this mode of communication. There is no privacy for the deaf person or the caller. Either party may feel inhibited by the this lack of privacy; this factor was also expressed as a contributing component in the discomfort of the previously mentioned interpreters. The interpreter is required, in some cases, to have signing skills or to be familiar with the oral deaf person's speech. These problems make this technique unacceptable in many situations.

2.1.2 Tactile Codes

A step up in independence for the oral deaf is a method that involves using the phone speaker as a vibratory-tactile device [Jamison,1974]. The deaf receiver lightly rests a finger on the handset's speaker and can detect the tapping on the mouthpiece of the caller's handset. An understanding must be reached regarding an encoding method that will be used in this means of communication.

The easiest code restricts the conversation to a series of yes/no questions that the deaf person asks the caller. Responses are generally coded as one tap for no, two for yes, three for repeat. This clearly places a serious restriction on both the rate and subject matter of the communication. This mode is perhaps most applicable in an emergency situation when the deaf person needs only to know that an urgent message has been successfully received.

A common encoding method in other telecommunication applications is Morse code [Jamison,1974]. Morse code consists of a series of dots, dashes, and pauses that can easily be adapted to tactile coding; one tap is a dot and two taps is a dash. This method unfortunately requires that the sender know Morse code. Morse code is also very slow; using the established average word length of 50 bits [Chester,1983], and assuming a distinguishable tapping rate of 75 taps per minute, communication rate will be just 3 words per minute.

The Y/N and Morse coding techniques presented are intended to represent the extremes of what could be accomplished using tactile coding. Specialized codes could be established by individuals who communicate frequently, but these would also be slow and inconvenient.

While the tactile coding method is inexpensive and has afforded the deaf person some measure of independence, the actual cost has been high. The non-impaired communicator must relinquish all conversational control, learn a code, or choose not to converse with the impaired person. All but the most committed individuals choose the latter.

2.1.3 Teletypewriters (TTYs)

A breakthrough occurred in independent telephone communication for the deaf when Robert Weitbrecht, a deaf physicist, invented the acoustic coupler modem in 1964 [Bellefleur,1976]. This invention allows two-way communication with teletypewriters over a standard telephone line. The manner of communication is simple; what is typed on one machine is reproduced on the other.

The earliest devices were both expensive and cantankerous; new TTYs cost as much as an automobile in 1965. 1968 saw the formation of Telecommunications for the Deaf, Inc. (TDI), a non-profit corporation that reconditioned TTY-33s (Figure 1), TTY-37s and TTY-47s being discarded by major telegram communication firms. Sold to the deaf at cost, a bare bones installation of a TTY costs approximately $500. These mechanical devices are large and noisy, and spare parts are no longer being manufactured, making them increasingly unsuitable for personal use [Levitt,1981].

More recently, electronic devices have been designed or adapted from the computer terminal industry to emulate TTYs. These Telecommunication Devices for the Deaf (TDDs), as they are called, are much smaller than their predesessors; many are battery-powered and fully portable (Figure 2). Displays are either electro-luminescent or typed copy. Some TDDs have message buffers that allow the user to compose a message and send it at the maximum transmission rate. These devices were originally more expensive than TTYs, but they now offer greater reliability, compact size, and low maintenance for about $600 [Vanderheiden,1978].

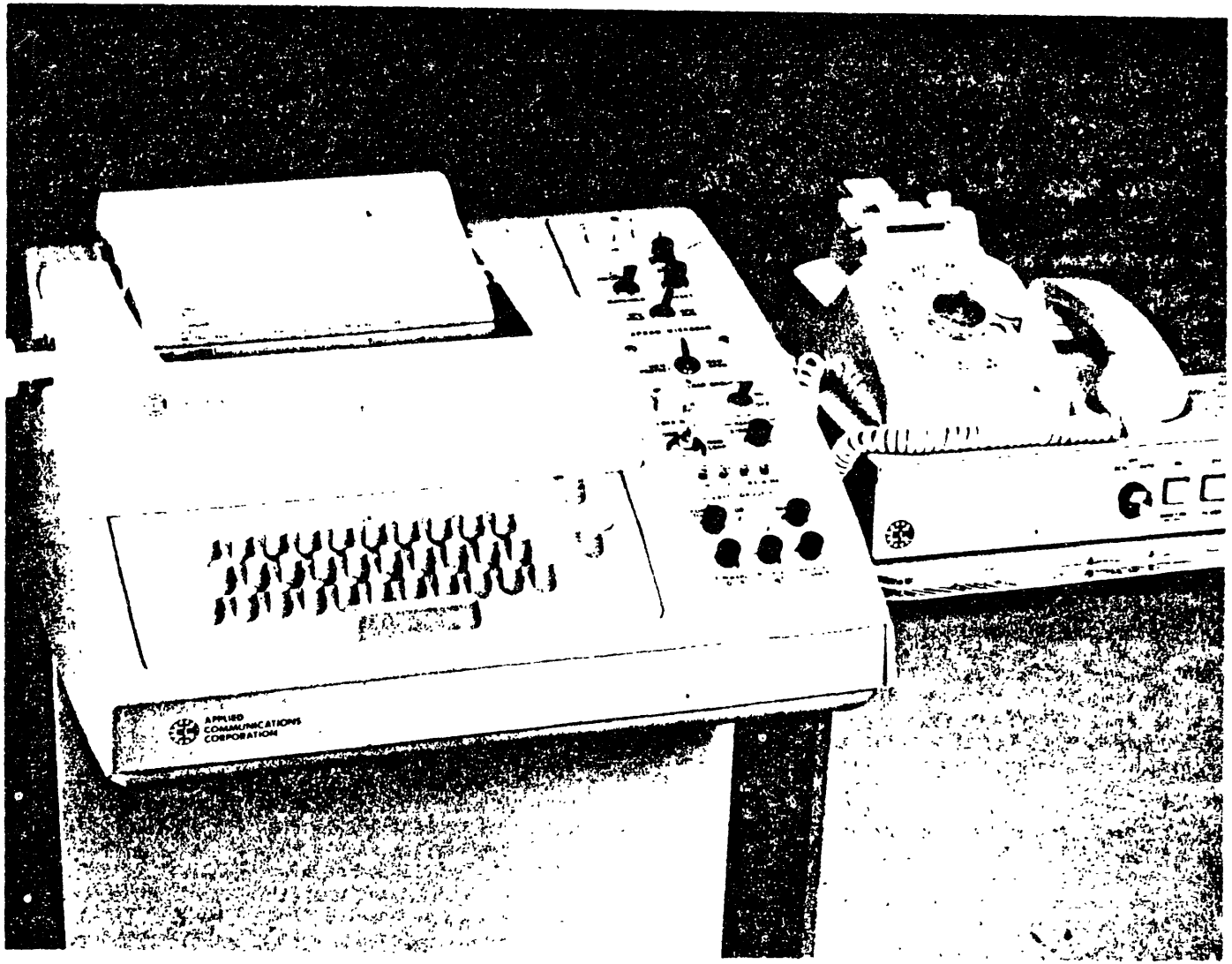

Figure 1. A TTY-33 adapted for use by the deaf.

Figure 2. An example of a TDD.

This family of devices, although a vast improvement over any previous offering, is not without faults. It is estimated that the average telephone conversation using a TTY takes ten times longer than a verbal call [Bruck,1978]. Communication rate using a TTY is limited to approximately 60 w.p.m. either by user typing speeds or the protocol ceiling, ie. the maximum baud rate. TTY users have developed cryptic abbreviations, such as CU for "see you", and other methods for increasing communication rates. However, the Baudot, five level code adopted by the network does not fully utilize the telephone channel bandwidth and thereby limits the possible communication rate. These factors make communication using TTYs expensive.

It is not clear why the deaf community chose to retain the Baudot code rather than adopt the Bell 103 seven-level protocol chosen by the computer industry in the mid-sixties. Had this move been made in, say 1965, the few hundred TTYs that had already been purchased could have been converted back to standard ASCII code and the remainder of the deaf community could have purchased less expensive standard computer terminals. In fact, it could easily be argued that the TTY actually created another telecommunications gap for the deaf by denying them the use of their devices as standard computer terminals.

The TTY has not come close to reaching all of the deaf population. An extremely conservative estimate of the U.S. population that would benefit from a TTY would be the four million "severely hearing impaired". Estimates of the number of TTYs installed in the U.S. range from 40,000 to 100,000 [Bowe,1984].

The TTY has had a tremendous positive impact on the quality of life for the deaf population. In some areas, police, fire, and other emergency services are now required to have TTYs, making these public services readily accessible to the hearing-impaired. Due to the support structure that arose with the proliferation of TTYs, they are considered a necessity by some of the deaf population. Incompatibilites with more modern equipment and the penetration to less than 3% of the user population suggest that these devices are far from ideal.

2.1.4 Message Relay

Message relay service is a link between deaf and hearing individuals. The service involves the use of two telephone lines, one equipped with a TTY, and a human relayer. When a hearing-impaired person wishes to telephone a non-impaired person, they call

the message relay service and tell the relayer the phone number of the person they wish to call. The relayer then calls that person. The conversation is proceeds with the relayer typing on the TTY what the hearing person speaks, and reading the deaf person's reply from the TTY to the hearing person on the other, conventional phone line [Becker,1984].

This is an effective method of communication between deaf and hearing people, but it suffers from several glaring faults. There is no privacy; the relayer, most likely a stranger to both communicators, is witness to the entire conversation. It is also a very labor intensive means of communication, a relayer must be present at all times that the service is offered. This technique is currently offered as a volunteer service in only a few metropolitain areas, it clearly serves a very limited number of the deaf population.

2.2 Present Work and the Future

This section will give the reader an indication of the kinds of devices that may be the product of current research in the area of telecommunications for the deaf. Also described is a system currently being implemented to alleviate some of the problems with existing TDDs.

2.2.1 Touch-Tone Devices

In 1963, after about 15 years of development work, AT&T, Bell Labs, and Western Electric began to make the Touch-Tone [1] dialing option (Figure 3) available. This option required substantial changes on the part of the utilities, involving both local central office modifications and customer premises equipment. The Touch-Tone service has proven to be popular; by the end of 1976 about 70% of the Bell System's non-commercial lines supported Touch-Tone service [Joel,1982]. In 1984, virtually all telephone customers in the U.S. can subscribe to this service.

The signal transmitted upon striking a Touch-Tone key is referred to as DTMF (Dual-Tone Multi-Frequency). In this system, two audible tones are transmitted simultaneously, one from a "low" group (697, 770, 852, or 941 Hz.) and one from a "high" group (1209, 1336, 1477, or 1633 Hz.). There are thus sixteen possible combinations

[1] Touch-Tone is a registered trademark of AT&T.

of tones that can be transmitted. Twelve are provided and ten used in a standard Touch-Tone telephone; the highest tone of the high group is being saved for expansion [Joel,1982]. Since the tones produced are in the audio range, they are transmitted even after the phone connection is made. The opportunity of built-in data communications implicit in this feature was recognized soon after the method was adopted and has seen many commercial applications [Broomfield,1980].

Bell Telephone was the first, in 1968, to propose and prototype a communication device (Figure 4) for the deaf that used Touch-Tone coding [Nelson]. The sender encoded the message that was to be transmitted. The device presented the receiver with the last three characters that were entered. This device was not commercially produced, due in large part to the limited availability of Touch-Tone dialing at this time.

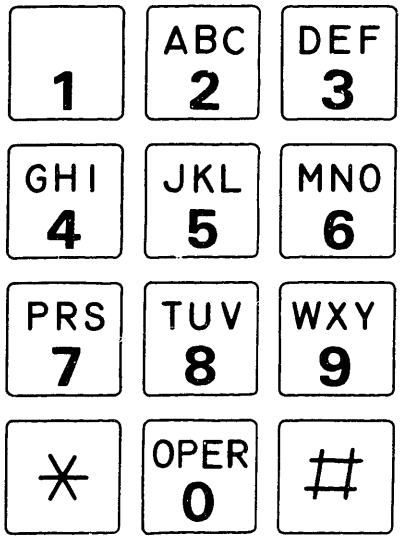

Figure 3. A diagram of the standard Touch-Tone keypad.

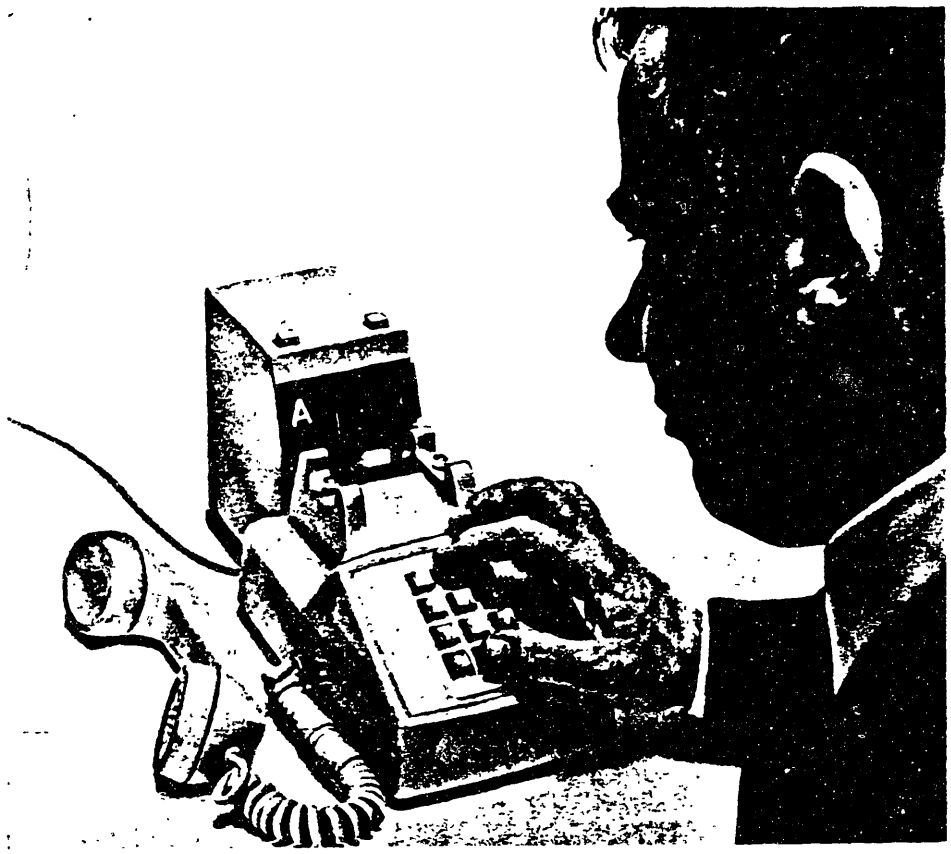

Figure 4. An experimental Touch-Tone device.

In the late 1970's several factors resulted in renewed interest in DTMF signals for reducing the telecommunication barrier between deaf and hearing individuals. The DTMF dialing system had become widespread, no longer confined to large metropolitain areas. The microelectronics field was also maturing; high-tech products were penetrating the consumer markets. A portable communication aid was now feasible that would make possible communication with any person having access to a Touch-Tone phone in any region of the country.

In 1981, Johns Hopkins University sponsored a nationwide search, for applications of personal computers to aid the handicapped, which exemplified this renewed interest. Two submissions to this contest dealt specifically with DTMF-based telecommunication aids for the deaf [Glaser,1981; Johnson,1981], however, neither device was made commercially available.

All of the proposed devices have required that the non-impaired sender encode the message being sent. That is, the sender must first strike the Touch-Tone key with the desired letter on it, and then strike some other key or keys to indicate which of the three letters on the key was intended [Nelson,1968; Wallingford,1974; Glaser,1981; Johnson,1981]. These codes, discussed in considerable detail in section 2.3.3, are clumsy and impose a workload on the message sender.

A device of this type has recently been introduced. While it remains to be seen whether this device will be a commercial success, its presence in the rehabilitation market will draw attention to the lack of effective devices to facilitate telecommunication between the deaf and hearing populations.

2.2.2 "Improved Telecommunications Aid for the Deaf"

An "Improved Telecommunications Aid for the Deaf" recently announced by researchers at the University of Texas at Arlington is a device that attempts to perform the same function as the device presented in this thesis. Their approach will be discussed in more detail in Section 4.6 when the reader has sufficient background about the problem to understand the techniques used. At that point the method they propose can be compared and contrasted to the algorithm developed herein [Shennib, *et al* 1984].

2.2.3 DEAFNET

DEAFNET is a computer networking system that supports both Baudot, five-level code TDDs and standard ASCII terminals. This project was started in 1978 at Southwest Research Institute (SRI) International of Menlo Park, CA. Slated to be running in the 20 largest U.S. cities by 1985, DEAFNET will help solve the problem of TDD protocol incompatability [Anon.,1984].

DEAFNET will work like a computer mail utility. Anyone having access to a TDD or an ASCII terminal can use the system to send, store, and receive messages. The system will also allow access to numerous nationwide databases, such as airline schedules and UPI news stories.

Although the technical barriers have been overcome, the tasks that remain are formidable: raising the necessary capital, training local leaders, and arousing sufficient public interest to get the system in place. The system currently runs on a VAX minicomputer, putting cost of an installation that supports 150 users at over $15,000. SRI is also investigating the possibility of using smaller, less expensive microcomputers as DEAFNET servers.

2.2.4 Speech Recognition Device

SRI International is currently working on a device that will allow a non-impaired person to simply "talk" to the deaf. The device uses a large-vocabulary, isolated-word speech recognizer to "listen" to the non-impaired speaker. The recognized text is then transmitted to the deaf user via visual display. The deaf person responds either by vocalizing, if sufficiently comprehensible, or by using a high-quality speech synthesizer [Anon.,1983].

This system is very appealing in that the nature of a conversation conducted in this manner is very similar to that of a phone conversation between two non-impaired individuals. However, these benefits can be explored only on a laboratory prototype system since this device is not yet technically feasible. Speaker-independent isolated-word recognition systems are currently limited to vocabularies of less than 100 words; the SRI proposal presumes a 5000 word system [Bernstein, *et al* 1984]. A system capable of recognizing 5000 speaker-independent words is a few years distant, and possibly additional time would be required for it to become financially viable.

While this system may be nearly ideal in terms of its user interface, the speech recognition technology required for its successful implementation is "on the horizon," a place it seems to permanently occupy. This waiting for technology, as a generic problem with rehabilitation products, will be discussed in more detail in the following section.

2.3 Need Identification Regarding Telecommunication Aids

This review has revealed many problems with the current state of telecommunications for the hearing-impaired. The survey of current research topics in this area shows that many of these problems are being addressed. Several additional elements must be included to complete this investigation.

2.3.1 The Rehabilitation Product Marketplace

One frequently neglected factor in the discussion of communication aids, or in fact, any rehabilitation product, is the nature of the rehabilitation product environment. The economic forces acting on and within this environment are different from those of many other consumer markets. Short product runs, poorly defined user needs, public funding for purchases, and numerous other factors combine to establish a context where ordinary product design practices frequently don't apply.

Rehabilitation product design is not a lucrative field; a manufacturer is not going to realize a profit unless the product is expertly targetted and marketed. It is particularly advantageous for a start-up product to be an assemblage of standard consumer products. This allows the rehabilitation product to take advantage of both the warranties and economies of scale of the consumer product, and the service network of the established manufacturer. For computer-based devices such as communication aids, this practice allows much of the actual function of the product to be implemented in software, simplifying development work and product improvement [Rosen *et al*,1983]. Finally, the prototype runs can be longer, allowing the marketplace to pinpoint the need that a product is (or isn't) meeting.

Another advantage of this system of product development is that it will frequently permit the development of an integrated system. For instance, a single microcomputer could easily function as a TDD, a standard ASCII computer terminal, and a Touch-Tone

decoding communication device. This system would be far more capable than any system currently available for deaf telecommunications, and could also serve as a standard microcomputer when not being used for communication [Levitt,1981; Rhodes,1982].

Unfortunately, rehabilitation products often have slightly different performance requirements than their consumer market counterparts. For example, while there is little reason for concern about power consumption in a commercial speech synthesizer, it becomes critical when the synthesizer is incorporated into a battery-powered portable speech aid for the non-vocal. The relatively small rehabilitation marketplace exerts very little market pull for products to satisfy its peculiar needs. Thus the handicapped are often left waiting for products that are simply not profitable to manufacture.

2.3.2 Need Identification

It is clear that telecommunications between the non-impaired and the deaf is difficult with any current system. While there may be many ways to facilitate this communication, it is vital that these techniques take into account certain general characteristics of the non-impaired population. In short, a non-impaired sender is unlikely to use any system of communication that is difficult, slow, or significantly imposing in any other way.

The Touch-Tone telephone keypad is a viable method of text entry accessible to virtually any non-impaired conversant. Directly entering text using the alphabetic coding resident on the Touch-Tone keypad results in a loss of information, since any transmitted code has three possible meanings. Previously proposed devices have used encoding schemes that require the sender to compensate for this loss of information by entering multiple keystrokes to specify a single character. These schemes are not intuitive, and slow the transmission rate.

The Touch-Tone text extraction algorithm permits direct text entry on the keypad, allowing the sender to transmit messages in a fast, intuitive manner. The algorithm compensates for the loss of information using the inherent redundancy of the English language. This method of text entry is very similar to typing, a task with which virtually any sender will be familiar. The sender needs only to learn a single new rule, how to produce the character **space**, before communication is possible.

2.3.3 Experimental Verification

An experiment was conducted to determine the effects of various methods of entering text using the Touch-Tone keypad on the speed of text entry, learning difficulty, and error rate. As was mentioned in Section 2.2.1, each technique that has been previously proposed for the entry of English text on the Touch-Tone keypad has required the sender to enter an additional keystroke or keystrokes to indicate which letter on the Touch-Tone key was desired. One method requires that after typing the key on which the desired letter appears, the sender must type the **1**, **2**, or **3** key to indicate whether the desired letter was the first, second or third letter appearing in the legend printed on that key. For example, to transmit an **S**, the sender would first strike a **7**, and then a **3** to indicate that the desired letter was the third letter on the **7** key.

The Touch-Tone text extraction algorithm allows the sender to enter text without the necessity of a second keystroke to specify which letter was intended. For example, to transmit an **S**, the sender would type the **7**. This method is easier to explain, easier to do, and requires fewer keystrokes from the sender.

An experiment was designed to investigate whether there were measurable differences in sender performance that could be attributed to the text entry techniques. Twelve naive senders were given a sample of text to enter on the Touch-Tone keypad of a standard desk-model telephone. Each sender entered the test sentences three times, each time using a different method of text entry. These three methods of text entry are explained in Appendix A.

The performance of the subjects, including speed of text entry, learning difficulty, and the number and type of errors, was monitored in the following manner. The rule sets, reproduced in Appendix A, were presented in balanced order, to minimize bias due to learning effects. After each rule set was read, the subjects were given a trial sentence to produce with the rule set prior to the test sentences. Subjects were closely watched for errors in their entry of the trial sentence; rules relevant to the error were repeated verbatim. This trial sentence was repeated until the subject felt comfortable with the rule set.

During the testing period, errors were assessed by the test administrator and informally verified with the subject. The test administrator sat within viewing distance of the subject, timing the subject and watching for errors. At the end of each timing

session, the subject was informed of the frequency and nature of their errors. In cases were the error resulted in an additional or reduced number of keystrokes, the time for the sample was adjusted accordingly. The number of errors was not formally recorded; the method of error detection was not sufficiently rigorous to merit more than anecdotal discussion.

The results of the experiment were tabulated by taking the mean and standard deviation of the text production times across subjects, and performing a Student's t-test on the results to establish significance of any observed differences. Table 1. gives a summary of the results; the raw data and an example calculation appear in Appendix B. The data indicated that the difference in the mean times between the single-keystroke method and the other methods was highly significant. (The t-values indicated that the probability of the difference in the means being due solely to chance was < 0.1%) When subjects were using the single-keystroke method they entered text 1.8 times faster than when they were using the quicker of the multiple keystroke methods. The text transmission rate for the single-keystroke method was 11 words per minute.

The error rate was lowest for the single-keystroke method. The only observed errors using this method were errors in transcription, whereas subjects using the multiple-keystroke methods tended to make errors in encoding the letters.

The subjects appeared to have more difficulty learning the multiple-keystroke coding rules. They repeated the trial sentence more often, and asked the test administrator more questions about the rules. Finally, when asked which rule set they liked best, all twelve subjects said they preferred the single-keystroke method over the others.

	Rule Set 1	*Rule Set 2*	*Rule Set 3*
Mean Time	2:53.6	3:15.8	1:34.9
Standard Deviation	24.7	37.9	18.6
Rate (wpm)	6.0	5.3	11.0

Table 1: Data Summary from Production Rate Test.

This experiment verified that senders performed better using the method of text entry allowed with the Touch-Tone text extraction algorithm. The technique was also unanimously preferred by the senders. The task remaining is to convert this transmitted string of digits into readable text.

3. Qualities of English Text

Peculiar qualities of English text make it possible to perform text extraction. These qualities can be demonstrated using methods derived from information theory and analogies from problems similar to the text extraction problem. A number of statistical studies of English text have been done that are useful in devising and analyzing various solutions.

3.1 Information Theory

This section presents only those aspects of information theory that will be useful in this thesis. The reader is urged pursue this topic through the references for this section.

Information theory as a scientific discipline can be traced to a paper published by Claude E. Shannon in 1948, entitled *A Mathematical Theory of Communication.* Therein, Shannon supposes the existance of a message source producing messages of a known type, say English text. The output of this message source is to be transmitted over a noisy communication channel. He addresses the question of how to encode this message to achieve the fastest, errorless transmission by that channel. Terms Shannon presented in this paper will prove useful in discussing the recovery of information lost in the transmission of text messages.

Shannon demonstrated that the generation of English words and text could be approximated by a mathematical process. Were this true, correct English text could be produced by a machine. The various stages of his demonstration are reproduced below.

Suppose that a box containing 27 cards has been prepared. Printed on 26 of the cards, one letter each, are the letters of the alpahbet. The last card is totally blank and represents the **space** character in written English. A string of characters is generated by repeating the following steps:

1. draw a card from the box,
2. write the character on the card next in the string, and
3. put the card back in the box.

Each letter in the box has the same probability of being chosen. Shannon called the result of this process a zero-order approximation to English text. The following sample of pseudotext, from Shannon [1948], was produced using this zero-order approximation:

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD QPAAMKBZAACIBZLHJQD

Clearly this is not a very accurate model of English text. There are far too many Xs and Js, not enough Es or Ts, and the tokens, ie. sequences of characters occuring between two delimeters (in this example, **space**), are coming out too long, suggesting not enough **spaces**.

One readily apparent way to improve the model is to choose the letters so they have the same probability of occuring in the string as in some representative sample of English text. These probabilities, referred to as letter frequencies, are presented in Appendix C.

Suppose a new box has been prepared containing 10,000 cards. A research assistant prepared the cards by looking at 10,000 written English characters in a running sample of text and writing each character on one card. Given a sufficiently representative sample of English text, the letters on the cards drawn from this box will appear approximately in proportion to their letter frequencies. Text is produced by repeating the sequence of steps used to generate the previous sample. Shannon called the outcome of this procedure:

OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL

a first-order approximation of English text. This string looks a bit more like English text. Fewer "odd" letters like X and Q appear in the string and the token lengths are closer to normal. There are, however, more subtle problems coming to light. For instance, experience suggests that there are very few occurances in English text of the letter H followed by the letter L (An attempt by the reader to quickly produce a word containing that letter pair will informally confirm this). This flaw in the model suggests that the probability of a letter occuring in the text string should be dependent on the previous letter in the string. These data, referred to as digram frequencies, appear in Appendix D.

Suppose now that 27 boxes have been prepared, 26 are labeled with the different letters of the alphabet, one is labelled **space**. The cards in these boxes were prepared by looking at 10,000 characters of English text, writing each character on a card and placing it in the box labelled with the character that preceeded it in the text. The string was generated by drawing a card from the box labelled **space**, writing the character on that card as the first character in the string, returning the card to the box, and then repeating the following steps:

1. draw a card from the box labelled with the last character added to the string,
2. write the character on the card next in the string, and
3. return the card to the box.

Shannon called the string generated this way, using digram frequencies, a second-order approximation of English text. His example:

```
ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE
TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE
```

This approximation technique has actually produced several English words and a number of pronouncable tokens.

Shannon extended this procedure once more to generate, using trigram frequencies (Appendix E), an example of third-order approximation text:

```
IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF
DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE
```

While the improvements are becoming more subtle, they are present. For instance, there are no tokens lacking vowels or comprised of single consonants.

Shannon presents a few additional examples based on words, however, his point is already well made. Clearly, if this technique for the generation of text were extended to include enough of the previous characters, fewer and fewer non-words would be produced. A machine can produce text that is arbitrarily close to written English, however, in the limit, it will only reproduce the text used decomposed for the statistics [Campbell,1982; Pierce,1961].

3.2 Cryptography

While at first glance it would appear that the task of cryptanalysis is quite similar to Touch-Tone text extraction, closer examination reveals the similarity to be superficial. Each task involves extracting English text from a string of symbols that are known to have been generated in a systematic fashion from English text. The fundamental difference between the two tasks is the type of transformation that occurred when the text was encoded.

Cryptography involves what set theorists call a bijective mapping: the ciphertext was somehow uniquely transformed in the encoding. If the receiver knows, or can derive, the function used in the encoding process then the string has a deterministically reversible relationship with the plaintext it represents. The objective of cryptography is to make this reverse transformation difficult, but not impossible.

Touch-Tone text extraction involves an injective mapping; the symbols in the resulting Touch-Tone string can represent several different symbols in the original text string. There is no key that can be used to deterministically extract the original text from the encoded string, since the mapping process has permanently removed part of the information in the original string. Nonetheless, an examination of cryptography is useful in order to become comfortable with statistical treatments of English text.

3.3 Spelling Checking and Correction

Spelling correctors have been of practical interest to computer science ever since computers were first used to manipulate text files. One of the earliest articles on the subject appeared in 1960 [Blair], and the first application program was reportedly done in 1971 [Peterson,1980]. Spelling correctors are becoming more common on the microcomputer systems that are being used in small businesses. A recent issue of *Byte*, a common small systems journal, contained advertisements for no fewer than 15 spelling correction programs.

Spelling programs vary in the amount and type of information provided when a possible error is detected. A spelling checker simply produces a list of the tokens that it believes to be incorrect. The user then uses a text editor to locate and correct the errors in the document. A spelling corrector both detects misspellings and generates

a list of correctly spelled words that may have been intended. Spelling correctors are generally interactive programs that query the user during run-time about how to handle a detected error and make the appropriate changes to the text file.

Spelling programs employ various techniques to detect errors in text files; it is these techniques that are relevent to the problem of extracting text from Touch-Tone sequences.

3.3.1 Token Lists

An early technique for spelling error identification produced a list of each distinct token in the document. Recall from the information theory section that a token is a string of characters occuring between two delimiters. The most common delimiters are spaces, but various punctuation marks, such as commas, are also valid. A token, then, is a sequence of characters that is assumed to represent a word. A document of any appreciable length has many fewer distinct tokens than total tokens since many tokens are used repeatedly. For example, a document 10,000 tokens long will contain, on average, about 1500 distinct tokens. The list of distinct tokens produced by a program of this type enables a proofreader to view each token without the surrounding context and in the absence of distracting inferance. If a count of the number of times each token appeared in the document is kept, the list can be ordered to reveal certain classes of misspellings. For example, if the list is displayed with those items occurring least frequently at the top, typographical errors will tend to appear early in this list by virtue of their rarity. This obviously will not be the case if the author of a document consistently misspells a word. Extracting distinct token lists is a computationally simple but labor intensive technique for spelling error identification [Peterson,1980].

3.3.2 Probabilistic Spelling Checking

A more sophisticated technique for detecting errors takes advantage of the skewed probabilities of letter sequences occuring in English text. The data presented in appendicies C, D, and E illustrate this point, many particular sequences are infrequent, some never occur. A token occuring in a document is potentially misspelled if it contains one or more infrequently occuring digrams or trigrams. The TYPO program on UNIX [1] uses this method of error detection. It constructs digram and trigram frequency

[1] UNIX is a registered trademark of AT&T

tables while it extracts a list of the distinct tokens in the document. Then for each token in this list it computes an *index of peculiarity.* The index for a token is the root-mean-square of the indicies for each component trigram of the token. The index of peculiarity for a trigram, XYZ for example, is computed as:

$$\log(f(x|y)-1) + \log(f(y|z)-1) \quad - \quad \log(f(x|yz)-1)$$

This index is a measure of the probability that the token was produced by the same source that produced the rest of the text. Misspelled words tend to have high indicies of peculiarity. This particular method will perform well only on documents long enough to generate statistically significant tables of digrams and trigrams. It will also tend not to detect words that an author consistently misspells [Morris and Cherry,1975; McMahon, *et al* 1978].

3.3.3 Dictionary Look-up

Dictionary techniques automate the task that the human proofreader performs on the data produced by the distinct token extraction and probabilistic checking methods. The list of tokens from the document is compared to a list of correctly spelled words; any token appearing in the dictionary is presumed to be spelled correctly. If a token is not on this list it may be misspelled and should be brought to the user's attention as a possible spelling error [Peterson, 1980]. This is the most common type of spelling program; all of the systems available for microcomputers use this method of detection misspelled words .

The size and organization of the dictionary for a program of this kind is critical to how well it performs. A dictionary that is too small will not contain enough of the correctly spelled words from the document and will not make the user's task appreciably easier. It may also contain only commonly used words which are easy to inspect in the original list. If the dictionary is too long, it becomes more likely that a misspelled word will be mistaken for another correctly spelled word from the dictionary. A long dictionary also involves more search time to determine if a token is in the list, leading to slow run times and user impatience.

3.3.4 Speling Corecshun

Techniques for spelling correction provide some insight into Touch-Tone decoding; the two tasks are quite similar. Spelling correction routines receive as their input tokens that have been identified, usually by a dictionary technique, as misspelled words. Spelling correctors generally work by successively applying a number of well-defined transforms to their input and comparing the results to the absolute references of the dictionary and ultimately, the user.

The transforms applied to the tokens submitted for spelling correction are derived from studies of common spelling errors. The program generates other tokens based on the following hypotheses:

Two of the letters are transposed.

There is an extra letter.

One letter is missing.

One letter is wrong.

Each of these new tokens, called *candidates*, is searched for in the dictionary. Candidates that are successfully located in the dictionary are presented to the user as possible alternatives to the token identified as misspelled.

Other techniques for spelling correction have been proposed, and some commercial spelling correctors have incorporated more extensive hypothesis lists incorporating common phonetic mistakes, but this basic algorithm is common to all production spelling correctors.

3.4 Handwriting Recognition

Handwriting recognizers are devices used in place of keyboards for entering data; the motivation for this substitution is not entirely clear. Letters are printed, generally with a special stylus, on a tablet that is connected to a computer. The algorithms attempt to determine what letter the user has printed on the tablet.

To recognize a given handwritten letter, in this context called a pattern, it is first analyzed to produce a feature vector, which includes descriptions of stroke lengths, directions, and classification of stroke connections. This feature vector is compared to a data base of feature vectors corresponding to each letter of the alphabet. The result of

these comparisons is expressed as the probability that the pattern matches a particular letter in the data base. However, since the measureable qualities of handwriting are not very consistant, there is not sufficient confidence in these probabilities to say that the letter with the highest probability was the one intended by the author. Thus, handwriting recognizers must use additional algorithms to determine with sufficient confidence which letter was intended.

Consider, for a moment, the process by which humans read a sample of handwriting. There are many circumstances in which it is difficult to decipher a letter or word, especially when written by an unfamiliar hand. When confronted with this situation, repeatedly looking at the unfamiliar token, expecting to suddenly be able to identify it, is counter-productive. Instead, one examines the remainder of the sample to extract clues as to the identity of the token. Say, for instance, that an entire word is in question. The available clues include an approximation of the length of the word, the context provided by the comprehensible portion of the sample, and other examples of how the author draws particular letters and strings. These clues usually allow one to reach a conclusion about what word was intended. A handwriting recognizer attempts to emulate this process using a number of methods.

3.4.1 Syntactic Technique

The first method used is to establish a syntax of possible input strings such that when a string of patterns is received, the possible results are highly constrained. Methods for achieving this include varying the lengths of valid input strings (spaces are considered to be perfectly recognizable), using small vocabularies, and including highly differentiable letters in the strings. The syntax can also establish whether a word is valid given the previous word. While these techniques make syntaxical algorithms reliable, they are not applicable to Touch-Tone decoding where the syntax of valid input is: any English text that can be represented by the letters on the Touch-Tone keypad, a highly unconstrained problem.

3.4.2 Markov Methods

Markov methods are used in free running text recognition algorithms. This technique is very similar to the probability method described in the spelling program section. Letters are chosen to maximize the following quantity:

$$g(X,Z) = \sum \log P(X_i|Z_i) + \sum \log P(Z_i|Z_{i-1})$$

where:

P(X|Z) is the probability of character Z
given the feature vector X, and
P(Z|Z) is the probability of the character Z
given the previous letter Z (last).

This involves a large amount of computation. A word of average length, 4.74 letters, made up of 26 valid letters, requires $26^{4.74}$, or 48.3 million, additions to compute the word score, too large a task for today's computers [Shinghal,1983]. The computational complexity of this approach has prompted researchers to study algorithms aimed at reducing the number of possibilities that must be considered.

3.4.3 Dictionary Methods

As with the spelling programs, a more sophisticated method is a dictionary method. For text recognition, this method involves finding the maximum of g(X,Z), above, over all words in the dictionary of the correct length. These methods perform well, but generally suffer from high computational complexity [Shingal,1983].

3.4.4 Hybrid Algorithms

Perhaps the most interesting algorithms for text recognition are the hybrids. These algorithms combine Markov and dictionary techniques, achieving performance equaling those of the dictionary method with as much as a 75% decrease in computational complexity [Shingal,1983].

3.5 English Orthography

Current research and applications in fields encompassing such topics as spelling, human memory, and the reading process require a comprehensive database of English orthography. Large capacity computers have greatly eased the task of gathering significant samples of written English, which previously required tedious hand counting of word, n-gram, and letter frequencies. Samples of spoken English, although more difficult to compile, are necessary for the study of verbal phenomena; written and spoken English are quite different. Tables 2, 3, and 4 contain summaries of major corpora for both written and spoken English.

3.5.1 Word Counts for Written English

The first count of American words was begun by Thorndike in the 1920's. A massive count even by today's standards, it is especially staggering to consider it was completely hand tabulated. The Thorndike-Lorge count, published as *The Teacher's Word Book of 30,000 Words*, served as the standard source of word frequency data until the late 1960's when it was superseded by more sophisticated counts [Solso,1982].

In the early 1960's work was begun by Kučera and Francis at Brown University on another corpora of American printed English. This work was funded by the U.S. Office of Education. The text, frequently referred to as the Brown corpus, was prepared in a standard machine-readable format, and was released in 1964. This corpora was characterized by several features:

a classification into 15 genres of printed text

a large number (500) of fairly short extracts (2000 words)

a nearly-random selection of extracts within genres

These carefully documented characteristics have allowed researchers to compare certain language features across genres and perform other analyses previously impossible. This corpus has been the subject of a huge body of work in the field of English language research since its release [Johansson,1982].

The first written record of this work, again authored by Kučera and Francis, appeared as the *Computational Analysis Of Present-day American English* in 1967. Included in this book was a frequency-ordered list of the words appearing in the corpus. The most frequent 100 words were also analysed within genres. This book is currently the most complete record of adult written English word frequencies [Kučera *et al*].

A corpus of British English paralleling the Brown corpus was begun by Leech and Leonard at the University of Lancaster and completed by Leech, Johansson, and Goodluck at the University of Oslo and Bergen Centre [Johansson,1982]. This has yielded an additional dimension of possible comparision in the field of English language research.

Another prominent written English language corpus is the American Heritage Intermediate Corpus. Gathered by Carrol, Davies and Richman in 1971, it represents

text as it appeared in textbooks for grades 3 through 9 at that date. It, too, is extensive with five million words gathered from 1045 sources in 500 word samples. Frequency data is presented, in addition to simple total usage, by grade level, curriculum, and library areas [Carrol *et al*].

Researcher (date)	*Sample Size*	*Characteristics*
Thorndike & Lorge (1944)	~20,000,000 words	Magazines, books, juvenile books
Kucera & Francis (1967) (Brown Corpus)	1,014,232 words	15 categories of reading material
Carrol, Davies, and Richman (1971)	5,088,721 words	School books: Grades 3-9
Leech, et al. (LOB Corpus)	1,000,000 words	British English. Equivalent to Brown Corpus

Table 2: Summary of major written English word counts.

3.5.2 Letter Counts for Written English

While, historically, letter frequency counts have been of greater interest to cryptanalysts than other professions, this situation has recently changed. Scholars in the field of psychology have realized the significance of letter frequency in word recognition, memory tasks, and other language tasks. Computer science applications include letter and n-gram frequencies in spelling checkers, text recognizers, and anticipatory text generators for the physically handicapped. The first letter frequency counts, conducted in the 1930's, were largely concerned with cryptography. A number of excellent books primarily on the wartime applications of those counts, are available and recommended to those interested in the subject [Kahn,1967; Lysing,1936; Pratt,1939].

Larger scale letter counts that began to appear in the early 1960's had as a central theme the structure of the English language as it related to psychological issues. In 1969, Zettersten published *A statistical study of the graphic system of present-day American English*, the first letter count to utilize the Brown corpus. Subsequent work with letter frequencies included analysis of the Brown corpus for positional frequency and versitility of various size letter groups. A large body of work now exists, allowing a researcher to investigate a hypothesis without repeating a lengthy gathering of statistical background material.

Researcher (date)	*Sample Size*	*Characteristics*
Pratt (1942)	500 words	British English
Attneave (1953)	10,341 words	Newspaper
Underwood and Schultz (1960)	15,000 words	American novels, magazines
Zettersten (1969)	1,000,000 words	Kucera and Francis (K&F)
Solso and King (1976)	983,400 words	K&F, single letter
Solso et al. (1979)	983,400	K&F, digram and trigram
Mayzner et al. (1965)	20,000 words	positional frequencies
Solso and King (1976)	262,883 words	K&F, positional frequencies
Rubin (1978)	999,464 words	K&F, initial and terminal
Solso and Juel (1980)	897,453 words	K&F, digram, positional
Solso et al. (1982)	983,400 words	K&F, initial digrams

Table 3: Summary of Major Written Letter Counts.

3.5.3 Spoken Word Counts

The first major spoken word count was done by French, Carter, and Koenig in the late 1920's to determine the relative frequencies of speech sounds. In this study, 80,000 words, in average samples of 40 words, were transcribed from phone conversations. This work was criticized for the transcription method employed in gathering the samples: nouns were collected one week, verbs another week, adjectives and adverbs a third week, and other speech parts for shorter periods. This did not allow for reliable comparision of relative frequencies of words from spoken and written English.

In 1966, Davis Howes published a major count of spoken English based on 250,000 words from 40 speakers. This count was the most extensive ever prepared and satisfied a need that investigators of verbal phenomena had been plagued with for many years. The Howes count has been criticized for not being presented in a particularly useful format. The count has only been published in alphabetical order; extraction of a frequency list is presumedly left to the dedicated.

Examination of a frequency list from the Howes count prepared by such a technique reveals that the corpus is littered with artifacts of the method used to gather the data. The subjects, 20 college sophomores and 20 patients from a V.A. hospital, were free to talk about anything they pleased. If the subject became tongue-tied, they were prompted by questions like "what do you think of the political situation?" This led to observable responses about Cuba, Kennedy, and other topics peculiar to that period.

The most recent and sizeable count of spoken English to date was published in 1979 by Dahl. This corpus is based on over one million words taken from recorded psychoanalytic cases from all over the United States. Fifteen seperate psychoanalytic cases were used, and 15 sessions were transcribed from each case. The speakers for the samples include both the patients and psychoanalysts. The sessions took place in eight different American cities. There were a total of 8 female speakers and 21 male speakers (one analyst contributed two cases). This corpus was also influenced by the method of data collection, but in a much less historical manner. The words that are noticeably out of place in the frequency listing refer more to the psychoanalytic setting. Dahl notes in the introduction to the corpus that the word "hour" occurs more frequently than might be expected because it is used to refer to the psychoanalytic sessions [Dahl, 1979].

Researcher (date)	*Sample Size*	*Characteristics*
French, et al. (1927)	80,000 words	Phone conversations
Howes (1966)	250,000 words	Students and patients
Dahl (1979)	1,000,000 words	Psychoanalytic records

Table 4: Summary of Major Spoken Word Counts.

3.5.4 Letter Counts from Spoken English

Letter counts from spoken English are extremely rare. While it may be true that the differences between written and spoken English are not observable at the letter level, no proof of this was found in this literature search. A likely reason for the absence of letter counts from spoken English is the lack of applications for such data. Since phoneme counts are more useful for speech research, these are far more common.

The only published letter count for spoken English that was found was *Informal Speech* by Carterette and Jones [1974]. This work is based on a small sample of 15,000 words gathered at a staged party.

3.5.5 Interesting Data

Percent Tokens Accounted for	*Spoken (Dahl)*	*Written (K&F)*
50	42	133
75	183	1,788
80	270	2,827
85	436	4,572
90	848	7,955
95	2,243	16,217
98	5669	30,124
100	17,871	50,406

Table 5: Comparison of Token Usage.

4. The Touch-Tone Text Extraction Algorithm

The text extraction algorithm has evolved through several generations; the algorithm was progressively modified to produce better approximations of English text and address various problems that arose during its development.

4.1 Review of the Problem

A message, in the form of English text, is being transmitted by a person, hereafter called the sender. This message is to reach and be understood by another person, the receiver. The sender is encoding the message for telephone transmission by "typing" using the standard alphabetic legends on the Touch-Tone keypad (Figure 4). The task for the algorithm is to process the transmitted signal, a series of keystrokes from the sender, and present comprehensible text to a deaf receiver. When evaluating the output of various algorithms the term "comprehensible" will become more rigidly defined. Qualities of English text string representations that contribute to their being readable will be identified and an objective measure will be defined for evaluating the resulting algorithms. While the ultimate success of the device will vary as a function of experience, the receiver's aptitude for tasks involving language processing, and the redundancy in the communication of a given sender-receiver pair, this objective measure will allow an algorithm to be evaluated before being incorporated into a device.

4.1.1 The Keystroke Digit String

The task of Touch-Tone text extraction begins with a string of digits, one for each symbol in the text being transmitted. If the word THE is being sent, the digits 843 will be received (Figure 4). The * key is transmitted to signify space. Additional rules can be learned as they are needed: the 0 (or OPER) represents any prosic punctuation and the 1 key represents Q, Z, and the apostrophe. The string that follows could, with tongue in cheek, be called negative first-order Touch-Tone text.

```
4*9268*86*46*733*843*3944248466*28*843*278*687386(OPER)
```

This sample hardly qualifies as text; it consists of digits rather than letters. A major contributing factor in the construction of a readable text string is that the symbols be standard English orthographical units, ie. letters.

4.1.2 Zero-Order Touch-Tone Text

A negative first-order Touch-Tone string can be thought of as a sequence of possible letters. The initial digit of the sample, 4, is known from reference to the Touch-Tone keypad to be a G, an H, or an I. It is followed, with no ambiguity, by a space. The next digit is a W, an X, or a Y. And so on, each digit having associated with it three possible letters. A compact notation for the letters that may have been intended is the three possible letters printed in a column. A string of received Touch-Tone digits, represented in this way will be referred to as zero order Touch-Tone text.

```
G WAMT TM GM PDD TGD DWGGAGTGMM AT TGD APT MTPDTM.
H XBNU UN HN REE UHE EXHHBHUHNN BU UHE BRU NUREUN,
I YCOV VO IO SFF VIF FYIICIVIOO CV VIF CSV OVSFVO?
```

This text string is superior to the negative first-order text; not only does it consist of valid English characters, but it also contains two English words. While it is possible, with sufficient practice and time, to thread through such a string, finding the path or paths that produce English words, this is not a readable text string. The author became capable of doing this during the development of the text extraction algorithm, but found it an unpleasant and tedious task.

4.2 Probabalistic Text Extraction

4.2.1 First Order Text Extraction

The author first attempted an aspect of the Touch-Tone text extraction algorithm for a microprocessor project for Advanced Design Projects, a graduate level design course offered by the Mechanical Engineering Department at M.I.T. The project specified a fully operational DTMF decoder to monitor a phone connection and perform a minimal amount of processing on the transmitted keystrokes.

In the spring of 1983, when this work was done, monolithic CMOS DTMF decoder chips had been in production for little more than a year, and the cost was approximately $70 each. A survey of electronic trade journals and product registries indicated that no stand-alone commercial device was available at this time to perform DTMF decoding. There was no budget for the project, nor were projects intended to be exercises in hardware design, so the DTMF hardware was simulated as described below.

A DTMF generating keypad from a Western Electric telephone was modified to allow monitoring of the switch closures. The four row and three column switches of the keypad were buffered with operational amplifiers and connected to the parallel input port of a Rockwell Aim 65 microcomputer. The port was continuously polled to detect switch closures. The keystroke was debounced and decoded in software. The output, discussed in the following paragraph, was printed on the 20-character-width paper-tape printer of the Aim 65.

The software written for the Aim 65 was limited by the small memory capacity of this machine; a primitive Touch-Tone extraction algorithm was implemented. The text output could be called, again with reference to Shannon, first-order Touch-Tone text. The text was printed in a three-running-row format similar to that described above as zero-order text. However, the letters appearing in a particular column, associated with a specific Touch-Tone key, were printed in descending frequency order. For example, if the digit 4 was received, the letters G, H, and I were printed with the most common letter I in the top row, the next most frequent letter H in the second row, and the G at the bottom. Figure 5 shows the column of letters associated with each of the keys. The example sentence appears below.

```
I WAOT TO IO SEE TIE EWIIAITIOO AT TIE AST OTSETO,
H YCNU UN HN RFF UHF FYHHCHUHNN CU UHF CRU NURFUN.
G XBMV VM GM PDD VGD DXGGBGVGMM BV VGD BPV MVPDVM?
```

Six English words appear on the top line of the output, a significant improvement over the previous examples. The goal of an ultimate text extraction algorithm is recovery of the exact text intended by the sender despite any ambiguity in transmission. The results of this hypothetical technique could be presented as a single line of text. It is desirable for text approximations to present the receiver with readable text in a single line, even if additional lines are necessary for clarity. One major factor in the evaluation of a proposed algorithm is the "correctness" of the algorithm's best guess.

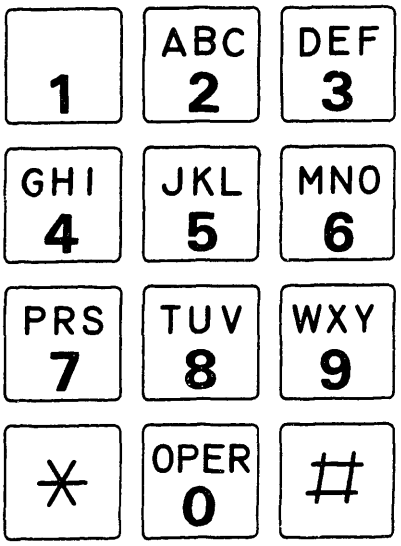

Figure 5. A diagram of the standard Touch-Tone keypad.

This device allowed trials of real-time communication with novice and seasoned receivers. The author, one of the latter, was generally able to decipher the intended message at or near the rate of sender production in the context of an established conversation. Isolated words were more difficult; longer ones exceedingly so. The novice users varied widely in their initial reaction to the device, from being appalled at the notion of reading the cryptic output to being delighted at the challenge. Approximately ten minutes of coaching was necessary before the most resistant were able to decipher a sentence without assistance. No difficulty was mentioned by any of the senders regarding using the Touch-Tone keypad for text entry.

This project lent some credibility to the possibility of a single-keystroke Touch-Tone telecommunication aid for the deaf. While no formal data was acquired, the device demonstrated that sufficient information was being transmitted for a conversation to be conducted with a experienced receiver. The trials with novice receivers suggested that a better approximation of English was necessary to reduce the initial alienation that the odd looking "text" evoked in some receivers.

4.2.2 Algorithm Scoring Program

While the performance of an algorithm can really only be evaluated in use, it is clear from the previous examples of algorithm output that qualitative assessment of output quality is possible. Quantitative measures of those aspects of algorithm output that were contributing to readability were identified. This allowed algorithms to be evaluated at an early stage of their development. Numerous tools for evaluating and testing algorithms were developed which greatly simplified these tasks.

The evaluation of an algorithm was originally a process of manually entering digit strings of known words for the program to decode. This was later partially automated with a program called DIG. DIG processed a text file into a digit file which was then used instead of keyboard input. Manual inspection of the results was required to evaluate the algorithm.

A program was written to automate the task of comparing the output of an algorithm to the text used to generate its input. This program, SCORALGO, recorded the frequency with which the algorithm correctly deciphered its input. This provided a means of objectively measuring one critical aspect of the performance of an algorithm. Ultimately, other factors must be taken into account in this evaluation, but the scores from this program were extremely helpful.

4.2.3 Second-Order Text Extraction

The text extraction techniques presented thus far have been quite simple. The algorithms have assumed no dependencies between subsequent possible-letters-intended. The information theory presented in chapter three indicated that this assumption is simplistic. Several techniques for incorporating digram statistics into algorithms for text extraction were developed.

Second order Touch-Tone text was not as easy to define as the lower order texts. To produce second-order text, Shannon had simply used digram frequencies to generate a string of characters (refer to section 3.1). This technique was adapted for the Touch-Tone string; the algorithm picked the character most likely to follow the previous one. For example, if a **4** was received as the first character of a Touch-Tone string, the algorithm would assume the previous character to be a **space**, consult a table to determine the frequencies of **space-G**, **space-H**, and **space-I**, and present the user with the character having the highest probability. The example text string, when processed in this fashion, becomes:

```
I WANT TO IN SED THE FYIGAITHON AT THE ART OVSETO.
```

This example shows several improvements over the top line of the first-order text. There are now seven English words in the output text. Two other tokens, while not actual English words, are at least pronouncable. The grammatical form of the sentence has become more clear. For example, words immediately following occurrences of THE can be thought of as noun phrases. This simplifies the deciphering task by constraining the number of words that must be considered for these positions.

The most-probable-digram-rule produces a single line of output. A representation of the other possible characters is desireable to allow one to decipher a poorly-approximated word. For example, the fourth word in the previous example, IN, is

obviously incorrect from the grammatical context of the sentence, since two prepositions never appear in conjunction. Unless the alternate characters are also presented, it is difficult to decipher this word.

There are actually several valid ways to generate these additional possibilities for user presentation. One method is to display, in columnar format, the other two possible characters in the order given by their probability of occurring after the most likely previous character. Consider this rule for the first character of this string. Of the G, H, and I, the I was found to have the highest frequency of occurance following **space**, the H was second, and the G last. Therefore, applying this rule, the three characters would appear as:

```
I
H
G
```

The next character is known to be a space. The rule is applied again to the 9 key:

```
I W
H Y
G X
```

The rule is now applied to the 2 key. At this point, the character positions are determined from the probability of A, B, and C occurring after the most likely previous character, the W. The string becomes:

```
I WA
H YB
G XC
```

The result of completing this process on the string:

```
I WANT TO IN SED THE FYIGAITHON AT THE ART OURETO
H YBMU UM HO PFE UIF DWHIBGVINM CV UIF CSU MTSDVM
G XCOV VN GM RDF VGD EXGHCHUGMO BU VGD BPV NVPFUN
```

Looking at the output from this technique, it does not appear that much has been gained by displaying the additional lines. Two words in the processed text remain virtually inscrutable. The two new lines seem not to give even a clue to their identity. The lower two lines may even be detrimental: their presence adding only visual confusion.

Given the low probability of some of the possible letters, it may be reasonable not to display them at all. This would give the algorithm freedom to create what may be more readable text. For example, the second line could display the highest probability string given the second most likely initial letter. The third line could do the same for the least likely initial letter, or perhaps the second most likely string using the most likely first letter.

The approaches thus far have assumed that the Touch-Tone string needed to be processed as it is being received. This assumption has been overly constraining, and has prevented the algorithms from using all the information available in the incoming string. In particular, the second-order text extraction algorithm virtually ignores the information content of the terminating spaces that delimit words. It is perfectly acceptable to buffer the input for a short period of time before displaying it, which proves to have many advantages.

4.3 Token Oriented Analysis

We have been considering the incoming string as a *series* of possible characters. It is also possible to view the problem at a higher level, i.e., analyze the incoming string as tokens separated by the delimiter **space**. When keystrokes are being received, any sequence of digits occurring after one space and prior to the next can be considered to represent an English word. Given a digit string of length N, with each digit representing one of three letters, there are 3^N possible letter strings. This decomposition is illustrated in Figure 6. Our task becomes one of determining which of the possible letter strings is the correct word (Figure 7).

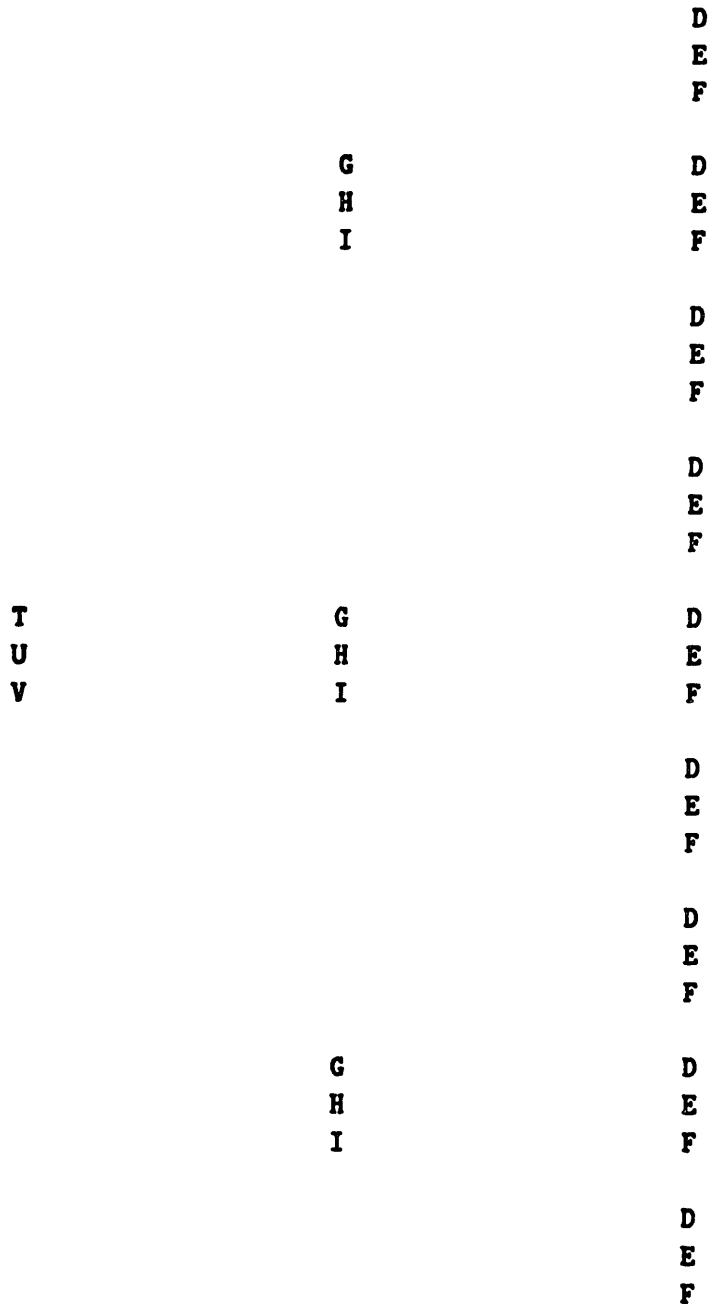

Figure 6. Graphical Representation of Possible character strings for 843

```
WAMT   WANU   WAMV   WANT   WANU   WANV   WAOT   WAOU   WAOV
WBMT   WBMU   WBMV   WBNT   WBNU   WBNV   WBOT   WBOU   WBOV
WCMT   WCMU   WCMV   WCNT   WCNU   WCNV   WCOT   WCOU   WCOV
XAMT   XAMU   XAMV   XANT   XANU   XANV   XAOT   XAOU   XAOV
XBMT   XBMU   XBMV   XBNT   XBNU   XBNV   XBOT   XBOU   XBOV
XCMT   XCMU   XCMV   XCNT   XCNU   XCNV   XCOT   XCOU   XCOV
YAMT   YAMU   YAMV   YANT   YANU   YANV   YAOT   YAOU   YAOV
YBMT   YBMU   YBMV   YBNT   YBNU   YBNV   YBOT   YBOU   YBOV
YCMT   YCMU   YCMV   YCNT   YCNU   YCNV   YCOT   YCOU   YCOV
```

Figure 7. The letter tokens possibile for digits 9268.

This task bears a distinct similarity to that of the spelling correctors and handwriting recognizers presented in chapter two. The possiblities of Figure 7 consist of 3^N-1 incorrect words and *one* "correct" word; "correct" meaning the word that was intended. In terms of English spelling, however, there may be more than one "correct" word. In the example sentence, the Touch-Tone sequence 4-6 occurs as a digit string. The list of possible words that this string could represent (Figure 8.) contains *two* very common English words. This is an example of what will hereafter be referred to as a *collision*: a case in which a single digit string represents more than one valid English word.

```
GM     GN     GO <--
HM     HN     HO
IM     IN <-- IO
```

Figure 8. Example of a ''collision'' (46 is both GO and IN).

This concept of identifying the correctly spelled letter string was central to development of the algorithm. It is, in fact, the unambiguous delimiting of letter strings by the character **space** that makes the final algorithm successful. The potential for error due to the incidence of collisions, however, must be examined.

4.3.1 Collision Frequency Check

The problem of collisions was disconcerting since frequent collisions indicate that further information, perhaps an extra keystroke, may be necessary to signify the intended letter. A program was written to investigate how frequently collisions would

arise as well as how many words would generally be colliding at a single Touch-Tone digit string. The program reads a list of words and translates the words into Touch-Tone codes as a sender would when entering text. The words and their associated digit strings are then sorted by digit string and printed. An excerpt from this list, showing how collisions appear as adjacent items, is given in Table 6.

	Digits	*Word*
	44	HI
	443464	HIDING
	4444	HIGH
	446	HIM
	4467353	HIMSELF
	447	HIS
	4475	GIRL
	44757	GIRLS
	448	HIT
	4483	GIVE
	44836	GIVEN
	44837	GIVES
	448464	GIVING
	4523	GLAD
⇒	46	IN
⇒	46	GO
	46243	IMAGE
	46243368	INCIDENT
	4624463	IMAGINE
	463333	INDEED
	46367628466	INFORMATION
	4637	GOES
	46464	GOING
	4653	HOLD
	4653464	HOLDING
⇒	4663	GONE
⇒	4663	GOOD
⇒	4663	HOME
	4663129	GOOD-BY
	466334283	IMMEDIATE

Table 6: A portion of the DIGIT collision check list.

A list consisting of the 1397 most common words spoken in the Dahl (1979) corpus, accounting for over 93% of the tokens spoken in the sample, was processed

by this program. The results, appearing in Table 7, were encouraging; over 90% of the words from the list had distinct Touch-Tone codes. Also encouraging was the occurance of only three "higher-order" collisions where more than two words had the same Touch-Tone representation. This exercise suggested that if an English word that can be represented by a received digit string is identified, there is a high probability that that is the word the sender intended.

Number of words processed 1397		
Collisions:		
Two types involved 46	words.....92	
Three types involved.....4	words.....12	
Total 50	words 104	

Table 7: Data from DIGIT collision check program.

4.3.2 Dictionary Method

After running the DIGIT program on the Dahl corpus, it was clear that one technique that could be used to recognize 90% of the incoming text would be a dictionary containing 1500 or so of the most common words used in spoken English. This would be computationally simple, and it is likely that many conversations would be understandable.

The problems with this method arise with words not contained in the dictionary; these could not be presented in a readable way using only dictionary lookup techniques. Many of the *content* words of English fall outside of a 1000-1500 word dictionary. If most of the content words of a sentence are unreadable, it is unlikely that sufficient context for conversation can be generated.

Other potential problems for a dictionary method are the words that were previously identified as "collisions". While there are not many of these, their occurrence may cause confusion about the meaning or even the structure of a sentence.

These arguments are not meant to discount the possibility of using a dictionary routine for Touch-Tone text extraction, only to suggest that it may be best suited as a supplementary technique.

4.3.3 Token-Oriented Second-Order Text Extraction

The algorithm was modified for second-order text extraction to take advantage of a delay between arrival of a digit string and display of its textual approximation. The fundamental advantage of this algorithm is the ability to analyze possible words with a *scoring function*, to find the best one. Using an approach similar to that of a probabalistic spelling checker, this algorithm maximizes the value:

$$\sum \text{Prob}(y|x)$$

for all the possible letter strings. The letter string having the highest score, assumed to be the intended word, would then be displayed. The text of the example sentence, when decoded using a computationally feasible approximation to this algorithm, becomes:

```
I WANT TN IN SEE THE FWHICITHON AT THE AST OURETN
```

While this text is not greatly improved over the previous example of second-order text, the method provides an ordered list of possible words that can be displayed as additional lines in the display. If the second and third highest scoring words are displayed also, the output looks like:

```
I WANT TN IN SEE THE FWHICITHON AT THE AST OURETN
G YANT TO IO REE TIE FWHIAITHON CT TIE ART OUREUN
H WANU UN GN SED THD FWHICITHOM AV THD ASU OURETO
```

The additional lines provide useful additional information in this display. However, the two large words are still not readable, and useful alternate representations are not displayed. This example also exposes the problem of high-probability digrams dominating the words produced as possibilities. For example, the longest digit string is incorrectly decoded, beginning with a very rare letter sequence due to the high probability of the initial digram **space-F**.

Actually, it is not surprising that this scoring function produced erroneous results. The chance of an event comprised of several sub-events is known from probability theory to be the *product* of the probabilities of the sub-events, not the sum. For

example, the probability of two consecutive coin tosses coming up heads, 1/4, is the probability of heads on the first toss, 1/2, multiplied by the probability of heads on the second toss, 1/2.

The token-based second-order text extraction algorithm was accordingly modified to sort the possible words using the product of their digram probabilities, ie.:

This scoring function produced the following results:

```
I WANT TO IN SED THE EWHICITINO AT THE AST NTRETO
H YANT UN IO RED TIE EWHIAITHON CT TIE ART NTSETO
G WANU TN IM SEE TID EWHIAITINO AV TID CRT OURETO
```

The second-order text extraction algorithm now produces good approximations to short English words; longer words are still unreadable, even in sentence context.

4.3.4 Token-Based Third-Order Text Extraction

The task of defining third-order Touch-Tone text will prove to be as difficult as second-order. The situation is similar. It would once again be possible to generate a string of text by receiving a Touch-Tone digit, determining which of the three letters it could represent is the most likely, given the previous digram. Based on the vastly superior results of second-order token-based text extraction, effort was immediately directed towards third-order token-based text extraction.

The first implementation of this algorithm again maximized the sum of the trigram frequencies over the digit string. The results of this method:

```
I WANT TO IN RED THE EXHICITION AT THE AST OURETO
H YANT UN HO SED THF EWHICITION BU THF ART OUSETO
G XANT VO GO PED THD DWHICITION CT THD BST MUSETO
```

This technique had the same faults as the summing method had in the second-order case. Certain trigrams were dominating the generation of possible words despite the occurrence of infrequent trigrams in other parts of the word. For example, the frequent occurrence of the trigram **space-T-H** was dominating the possible words for the digit sequence **8-4-3** in the example sentence despite the subsequent occurrence of the extremely rare trigrams **T-H-F** and **H-F-space**. The first solution for this problem was

to compare the trigram value to a cutoff value before allowing the string to remain in consideration as a possible word. The results of this change:

```
I WANT TO IN RED THE EXHICITION AT THE AST OURETO
H YANT UN HO SED VID EWHICITION BU VID ART OUSETO
G XANT VO GO PED TID DWHICITION CT TID BST MUSETO
```

Although the performance of the trigram-based algorithm was already superior to that of the digram-based routine, it was assumed from the experience with the second-order algorithm that additional gains were possible by a multiplication-based scoring function.

The results produced using this technique were indeed significantly superior to those of the previous methods. The example sentence actually has very few obvious improvements:

```
I WANT TO IN RED THE EXHICITION AT THE AST OURETO
H YANT UN HO SED VID EWHICITION AU VID ART OUSETO
G WANU VO IN PED TIF EXHICITHON BU TIF APT MUSETO
```

The scoring program, working with a much larger sample of text, gives a more accurate impression of how much better the algorithm is functioning.

Using normalized trigram transition frequency data resulted in an additional improvement. This improvement is observable both in the results of the algorithm scoring program and in the lower lines of the example sentence:

```
I WANT TO IN RED THE EXHICITION AT THE ART OURETO
H YANT UN IN SED VID EXHIBITION AU VID AST MUSETO
G XANT VO GO REE VIE EXHICITHOM AV VIE APT OUSETO
```

Table 8 compares the performance of the various algorithms as processed by SCORALGO. Appendix F contains the list of words that was processed by the algorithm scoring program. Words appearing in italics were correctly deciphered.

Word Matched	Digrams (additive)	Digrams (multiplicative)
First guess	138	214
Second guess	97	127
Third guess	76	98
Fourth guess	63	77
Fifth guess	38	46
Sixth guess	21	31

Table 8a: Data from the SCORALGO Evaluation Program

Word Matched	Trigrams (additive)	Trigrams (multiplicative)	Trigrams (normalized data)
First guess	381	517	555
Second guess	200	262	252
Third guess	125	148	146
Fourth guess	102	106	101
Fifth guess	58	58	49
Sixth guess	35	26	40

Table 8b: Data from the SCORALGO Evaluation Program

4.4 The Hybrid Algorithm

That a hybrid algorithm would be the product of this work was actually known long before the preceeding account of the development of the algorithm would suggest. It was observed early on that many words, even common ones, are not highly probabilistic. A dictionary routine will recognize these words containing statistically infrequent letter compositions. The technique would also allow the user to establish, maintain, and bring into use during phone conversations, personal dictionaries. A personal dictionary would contain names of people and places, whatever words the deaf user wants displayed correctly.

The hybrid technique is actually two of the previous techniques running in parallel. When a digit string is received, a word searched for in the dictionary, and the third-order token-based text extraction algorithm is run on that string. When each routine is complete, one of two situations exist:

> The dictionary search has been successful and the trigram analysis has a list of tokens ready, or
>
> The trigram analysis has a list of tokens ready.

In the former situation, the dictionary word will be displayed first on the display, optionally followed by some of the probable tokens. If the situation is the latter, some combination of the probable words will be displayed. What should be displayed to the user will be determined after more communication experience with the prototype device.

It should be noted that any word that is correctly deciphered by the trigram routine does not need to appear in the dictionary. This effectively increases the size of the dictionary; approximately 40% of the words that were to appear in it can be omitted. Appendix F presents the 1397 most frequent words from the Dahl corpus; italicized words were correctly deciphered as the algorithm's first choice and can be omitted.

The example sentence after being decoded with the hybrid algorithm:

```
I WANT TO GO SEE THE EXHICITION AT THE ART OURETO
```

In a two line display format:

```
I WANT TO GO SEE THE EXHICITION AT THE ART OURETO
H YANT UN IN RED VID EXHIBITION AU VID AST MUSETO
```

And finally in a three line display format:

```
I WANT TO GO SEE THE EXHICITION AT THE ART OURETO
H YANT UN IN RED VID EXHIBITION AU VID AST MUSETO
G XANT VO IN SED VIE EXHICITHON AV VIE APT OUSETO
```

One additional possibility is that this deaf person frequents museums and has added this poorly decoded word to their personal dictionary, or possibly to a dictionary for conversations with this particular individual. The sentence would then appear:

I WANT TO GO SEE THE EXHICITION AT THE ART MUSEUM

It would be difficult for this message to be misinterpreted.

4.5 Discussion of Competitive Approach

As was mentioned in Section 2.2.2, researchers at the University of Texas at Arlington have recently published another approach to this problem [Shennib and Kondraske, 1984]. They also observe that a high percentage of commonly used words have unique Touch-Tone key sequences. Citing high memory requirements and recognition limited to the words appearing in the dictionary as significant disadvantages of a dictionary technique, they seek an alternative method.

They introduce the concept of a *syllable-like letter group* (SLLG), frequently occurring letter strings. From their description, these are derived in a manner similar to WRITE units developed by Goodenough- Trepagnier for efficient entry of text by non-vocal individuals [1982].

They propose to employ these SLLGs in the following manner: the incoming Touch-Tone string is searched for occurrences of these SLLGs, and they are connected to form words. A single-word example is given, it is reproduced here.

No mention is given as to how their example has been divided. The alternate decoding, using WRITE units, actually uses more frequent letter sequences than the word "engineering".

The problems that arise using this method of text extraction are the number of *collisions* among SLLGs (11 double collisions, 3 triples in WRITE-100 alone), and the lack of ways to determine where SLLG boundaries are located. This technique appears to hold some promise, but a more extensive demonstration and further explanation is needed.

5. Implementation

The hardware and software implementation of a prototype device comprised a large portion of the work involved in the development of the algorithm. Evaluation of competitive algorithms is tedious and time-consuming work, perfectly suited to a patient computer. In addition to its role in the development of the algorithm, this prototype will be duplicated and pilot-tested to obtain feedback from typical users.

5.1 Hardware Implementation

The first component that was considered in the specification of hardware for this thesis was the Touch-Tone decoder. Workable DTMF decoders are a recent development, and off-the-shelf hardware incorporating this function is uncommon [Ciarcia,1981]. Several devices incorporating Touch-Tone decoding are available, ranging in price from several hundred to several thousand dollars. Appendix G contains more specific information about currently available hardware including that used in the prototype.

The microcomputer used to implement the algorithm was an IBM Personal Computer [1] with a moderate amount of memory. This choice is not a crucial one; other choices could significantly lower the implementation cost. A notebook-sized computer could be used to implement a portable version, however, none of the modems currently built into these units have the Touch-Tone decoding feature. Several of the available decoder chips are available in low-power CMOS versions, so a battery-powered DTMF decoder is certainly possible.

5.2 Software Implementation

While it would be possible to walk the reader through the actual code of the program(s) written for this project, this would be of little benefit. The code that has been written was, in essence, a development system for the algorithm. That is to say that the code is not at all optimized for speed, compactness, or anything

[1] IBM is a registered trademark of International Business Machines, Inc.

other than a degree of flexibility in trying different approaches to Touch-Tone text extraction. Discussion in this section will center around major approaches to the problem of coding the algorithm. In addition, any special requirements for a section of code will be pointed out.

5.2.1 The Dictionary Storage and Look-up

The dictionary section of the algorithm had several functional requirements:

Search time must be reasonable.

Dictionary must be fairly compact.

Dictionaries must be readily extensible.

These requirements will be defined and discussed in some detail below.

The requirement for speed is clearly imposed by the real-time nature of the task of Touch-Tone text extraction. The program must be able to perform all operations on a digit token quickly enough not to miss keystrokes from the sender, ie. it is undesirable to code extensive buffering routines for the keystrokes to buy more time to process digit tokens.

The requirement for compactness stems largely from the nature of the data being stored in the dictionary. The words vary in length from 1 to 14 letters, with an average word length of 6.1 characters. A 1000 word dictionary, containing both alphabetic and numeric representations of each word, requires 14,200 bytes. If a simple form of data compression is employed, the dictionary can be represented in 6100 bytes.

The requirement that the dictionary be readily extensible stems from the need for the user to be able to add words to the base dictionary, add topic-specific and person-specific dictionaries on the fly, and to remove words and dictionaries as well. This requirement suggests that either the dictionaries be accessible in human-readable form or that the program include dictionary maintenance and creation utilities.

5.2.2 Method Employed for Implementation

The dictionary look-up routine for the text extraction algorithm was implemented as an indirect hash table. Hashing storage algorithms are used to provide quick access to large tables, using a hashing function to determine the storage location for a

particular key. Obviously, the hashing function must always compute the same storage location for the same key.

For the dictionary routine, the key is the digit string. When the program is initially run, the dictionary is read word-by-word from storage. When a word is loaded, the program constructs the digit string that would be entered on the Touch-Tone keypad to transmit that word. This digit string is passed to the hashing function, which calculates an integer value that falls within the dimensions of the hash table array. This is referred to as the hash table index.

Stored at each location in the hash table array are two pointers, one to a digit string the other to a character string. When the index of a key is returned, the program checks to see if the location `index` in the hash table is occupied by a value other than the initialized value. If it does not, the digit string and character string are copied into memory and pointers to their locations are stored in the hash table. If the hash table does contain a value other than the initialized value, one of two things has occurred.

One possibility is that the hashing function has returned the same index for two different digit strings. This is called a collision (not to be confused with a Touch-Tone collision). Hashing functions are designed to do this as infrequently as possible, but it may occur. When a collision occurs, the program generates a new index based on the previous index. This is then treated in the same way as the original index; the location `index` is checked for values other than the initialization value, etc. New indices are calculated until an unoccupied location in the table is located.

The other possibility in the case of Touch-Tone digit strings is that the same word is being stored a second time or that a Touch-Tone collision is being stored. For either of these conditions it is undesirable for the word to have storage allocated for it.

To distinguish between the case of a hash table index collision and a dictionary collision, when a position in the table is occupied, the current digit string is compared with the digit string. If this comparision is false, the collision is a hash index collision. If the comparision is true, the digit string already has a character string associated with it.

During program execution, when a digit string is received, it is necessary to determine if there is a dictionary word with which it corresponds. The digit string is passed to the hashing function which returns an index into the hash table. The digit

string is compared to the digit string at the location pointed to by the value in the hash table. If those strings are equal, the character string at the location pointed to by the value in the hash table is the dictionary word for that digit string. Otherwise, another index is calculated for the digit string and the process is continued until all possible locations where the digit/character string pair may have been stored are exhausted.

This has proved to be an expedient and robust method for dictionary storage and look-up. The major disadvantage of the method is the inability to associate and store more than one dictionary word with a digit string. Techniques to overcome this disadvantage are available and can be included if deemed necessary after conclusive pilot testing with experienced users. This description should aid any reader interested in interpreting the code of the algorithm presented in Appendix H.

5.2.3 Trigram-based String Generator

The portion of the program that generated the most likely character strings for a received digit string had the following requirements:

Must run quickly.

Storage space for possible strings must be small.

These two requirements are very closely related. The number of possible strings that need to be evaluated to establish which has the highest probability is 3^{length}. The program should be able to perform the maximization calculation on strings of up to 16 characters. This string alone would require the calculation of 43,000,000 scores if the routine was totally rigorous. This number of calculations and the space necessary to contain the results is excessive. The trigram-based string generator incorporates a number of techniques to reduce the number of calculations necessary to determine the most likely string and to minimize the number of string representations that must be maintained.

5.2.4 Method Employed in Implementation

The task for the string generator/scorer is akin to finding a needle in a haystack using only a pitchfork. It has been noted above that an exhaustive search of the possible text strings would involve an astronomical amount of calculation. Nonetheless, the maximum scoring string must be located with a high degree of confidence.

One way to satisfy these requirements was to determine a method for reducing the number of possible strings that needed to be scored. The actual method used was discovered while hand-calculating scoring functions.

It was necessary, at times, to do the scoring for a digit string by hand. This was an extremely tedious task. To avoid as much useless calculation as possible, the following technique for string scoring was adopted:

1. The first two digits were written as nine two-character sequences.
2. These nine possibilities were assigned probabilities from the trigram table.
3. Strings of zero probability were abandoned.
4. The third digit was added by concatenating each character it could represent with the strings remaining after step 3.
5. Scores were calculated based on trigram probabilities.
6. Low scoring sequences were abandoned.
7. And so on... unitl all digits have been included in the string.

This proved to be an effective method for limiting the amount of hand calculation required to score a digit string. Subsequent exhaustive scoring with the computer showed no cases where this technique failed.

The algorithm mimics this heuristic process by retaining only the top six strings after the addition of a digit. The technique works in the following way:

1. The first two digits are expanded to nine character strings and scored.
2. The six highest strings are retained.
3. Including the next digit produces 18 strings.
4. These are scored and those with the six best scores are retained.
5. Steps 2 and 3 are repeated until all digits have been processed.

The number of strings retained from step-to-step was determined by reducing the number until a decrease in performance was observed. In the next generation of the algorithm the number of possibilities retained will be a function of the digit string length; longer strings benefit from retaining more possibilities.

The algorithm incorporates another technique to minimize the chance of abandoning a string that would eventually have become one of the top scoring tokens.

The process outlined above is also run from the end of the digit string to the front. This technique frequently produces high scoring tokens missed by the left-to-right routine, generally those beginning with infrequent letter sequences.

Although the techniques employed thus far in executing the algorithm have been successful, additional work on the algorithm is planned. This work will be directed towards increasing efficiency, to permit its implementation on less powerful machines.

5.3 The Working Prototype

The working prototype has performed very well in the informal testing done so far; conversations have been both established and conducted using it. Hearing impaired users having average reading skills should have no trouble using the device.

Several aspects of the user-interface have not been fully implemented. The ability to edit personal dictionaries and manipulate them during conversations should increase both the accuracy and the useability of the device. Another obvious extension of the device is the inclusion of a speech synthesizer so that a non-oral deaf person can "talk" back to the caller.

Finally, the device must be evaluated by deaf people, under typical conditions. Local deaf organizations have expressed interest in the device, and there is no shortage of volunteers for device evaluation. The prototype will also provide convincing demonstrations of the potential benefits a production version would have for deaf people: totally independent telephone communication to virtually anyone.

Bibliography

Alberga, C.N. (1967). String similarity and misspellings. *Communications of the ACM.* **10**(5), 302-313.

Anonymous. (1968). Experimental device may extend use of Touch-Tone telephone to the deaf. *Bell Laboratories Record.* **46**(1), 34.

Anonymous. (1980). Contributions of technology to deaf and hearing-impaired individuals. *Rehab Brief.* **3**(11), 1-4.

Anonymous. (1983). Vocational rehabilitation with hearing-impaired clients. *Rehab Brief.* **6**(10), 1-4.

Anonymous. (1983). DEAFNET network links TTYs with standard computers. *Update.* (11), 2.

Anonymous. (1983). Device for deaf talks to standard phone. *Update.* (11), 3.

Becker, S. (1983). **Contact: deaf message relay training.**

Beker, H. and Piper, F. (1983). **Cipher Systems: the protection of communications.**

Bellefleur, P.A. (1976). TTY communication: its history and future. *Volta Review.* **78**, 107-112.

Bernstein, J., Becker, R., Bell, D., Murveit, H., Poza, F. and Stevens, J. (1984). *Telephone communications between deaf and hearing persons.* **Proceedings of the International Conference on Acoustics, Speech, and Signal Processing.** 26.7.1-4.

Beth, T. (1983). Cryptography: proceedings of the workshop on cryptography. *Lecture Notes in Computer Science.* (Goos, G. and Hartmanis, J., ed.). **149.**

Blair, C.R. (1960). A program for correcting spelling errors. *Information and Control.* **3**(1), 60-67.

Bourne, C.P. (1977). Frequency and impact of spelling errors in bibliographic data bases. *Information Processing and Management.* **13**(1), 1-12.

Bowe, F.G. (1984). Personal Communication.

Bozzuto, R.C. Jr. (1981) The universal translating modem. **Proceedings of the Johns Hopkins First National Search for Applications of Personal Computers to Aid the Handicapped.** 62-64.

Broomfield, R.A., Foxall, T. and Beirne, P. (1980). Making a data terminal out of a Touch-Tone telephone. *Electronics.* **53**(15), 124-129.

Bruck, L. (1978). **Access: the guide to a better life for disabled Americans.**

Campbell, J. (1982). **Grammatical Man.**

Carroll, J.B. (1971). Measurement properties of subjective magnitude estimates of word frequency. *Journal of Verbal Learning and Verbal Bahavior.* **10,** 722-729.

Carrol, J.B. and Davies, B.R. (1971). **Word Frequency Book.**

Carterette, R. and Jones, L. (1974) **Informal Speech.**

Chapanis, A., Ochsman, R., Parrish, R. and Weeks, G. (1972). Studies in interactive communication modes on the behavior of teams during interactive problem solving. *Human Factors.* **14,** 487-509.

Cherry, C. (1978). **On Human Communication: a review, a survey, and a criticism.**

Cornew, R.W. (1968). A statistical method of spelling correction. *Information and Control.* **12**(2), 79-93.

Dahl, H. (1979). **Word Frequencies of Spoken American English.**

Damerau, F.J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM.* **7**(3), 171-176.

Davidson, L. (1962). Retrieval of misspelled names in an airlines passenger record system. *Communications of the ACM.* **5**(3), 169-171.

Fitts, P.M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology.* **47,** 381-391.

Foulds, R., Baletsa, G. and Crochetiere, W.J. (1975). The effectiveness of language redundancy in non-verbal communication. **Proceedings of the Conference on Devices and Systems for the Disabled.** 82-86.

Gaines, H.F. (1956). **Cryptanalysis: a study of ciphers and their solution.**

Galli, E.J. and Yamada, H.M. (1967). An automatic dictionary and verification of machine-readable text. *IBM Systems Journal.* **6**(3), 192-207.

Galli, E.J. and Yamada, H.M. (1968). Experimental studies in computer-assisted correction of unorthographic text. *IEEE Transactions on Engineering Writing and Speech.* **EWS-11**(2), 75-84.

Gibler, C.D. and Childress, D.S. (1982). Language anticipation with a computer based scanning communication aid. **Proceedings of the IEEE Computer Society Workshop on Computing to Aid the Handicapped.** 11-15.

Glaser, R.E. (1981). A telephone communication aid for the deaf. **Proceedings of the Johns Hopkins First National Search for Applications of Personal Computers to Aid the Handicapped.** 11-14.

Goodenough-Trepagnier, C. and Rosen, M.J. (1982). Optimal language menu for a one-switch non-vocal communication device. **Proceedings of the Fifth Annual Conference on Rehabilitation Engineering.**

Goodenough-Trepagnier, C., Tarry, E. and Prather, P. (1982) Derivation of an efficient Non-Vocal Communication System. *Human Factors.* **24**(2).

Hanson, A.R., Riseman, E.M. and Fisher E.G. (1976). Context in word recognition. *Pattern Recognition.* **8**(1), 35-45.

Harmon, L.D. (1972). Automatic recognition of print and script. *Proceedings of the IEEE.* **60**(10), 1165-1176.

Howes, D. (1966). A word count of spoken English. *Journal of Verbal Learning and Verbal Behavior.* **5**, 572-604.

Jacobs, L.M. (1974). **A Deaf Adult Speaks Out.**

Jaffe, D.L. (1982). A computerized message system for the rehabilitation community. *8th Annual Computer Faire.* 107-110.

Jamison, S.J. (1974). Telephone communication for deaf people. **Proceedings of the Conference on Devices and Systems for the Disabled.** 58-62.

Joel, A.E. Jr. et al. (1982). Switching Technology 1925-1975. **A History of Engineering Science in the Bell System. 3.**

Johansson, S. (1982). **Computer Corpora in English Language Research.**

Johnson, A.B. and Hagstad, R.F. (1981). DTMF telecommunications for the deaf and speech-impaired. **Proceedings of the Johns Hopkins First National Search for Applications of Personal Computers to Aid the Handicapped.** 29-32.

Jurgen, R.K. (1980). Electronics in medicine. *IEEE Spectrum.* **17**, 81-86.

Kahn, D. (1967). **The Codebreakers: the story of secret writing.**

Knowlton, K.C. (1976). Method and apparatus for using pushbutton telephone keys for generation of alpha-numeric information. **U.S. patent 3,967,273.**

Kucera, H. and Francis, W. (1967). **Computational analysis of present-day American English.**

Kullback, S. (1976). **Statistical Methods in Cryptanalysis.**

Levitt, H. and Nelson, J.R. (1970). Experimental communication aids for the deaf. *IEEE Transactions on Audio and Electroacoustics.* **18**, 2-6.

Levitt, H. (1981). A pocket telecommunicator for the deaf. **Proceedings of the Johns Hopkins First National Search for Applications of Personal Computers to Aid the Handicapped.** 39-41.

Lysing, H. (1936) **Secret Writing: an introduction to cryptograms, ciphers, and codes.**

McElwain, C.K. and Evans, M.E. (1962). The degarbler – a program for correcting Machine-read morse code. *Information and Control.* **5**(4), 368-384.

McMahon, L.E., Cherry, L.L. and Morris, R. (1978). Statistical text processing. *The Bell System Technical Journal.* **57**(6), 2137-2154.

Martin, J. (1977) Pushbutton telephones and voice response. **Future Developments in Telecommunications.** 95-111.

Mayzner, M.S. and Tresselt, M.E. (1965). Table of single-letter and bigram frequency counts for various word-length and letter-position combinations. *Psychonomic Monograph Supplements.* **1**(2).

Meacham, L.A., Power J.R. and West F. (1958). Tone ringing and pushbutton calling. *Bell Systems Technical Journal.* **37**, 339-360.

Meyer, C.H. and Matyas, S.M. (1982). **Cryptography: A New Dimension in Computer Data Security.**

Morgan, H.L. (1970). Spelling correction in systems programs. *Communications of the ACM.* **13**(2), 90-94.

Morris, R. and Cherry, L.L. (1975). Computer detection of Typographical errors. *IEEE Transactions on Professional Communications.* **PC-18**(1), 54-64.

Nelson, J.R. (1968). *A Touch-Tone to visual letter decoder.* **MM-68-1232-3.**

Nelson, J.R. (1970). Experiments on the use of the Touch-Tone telephone as a communication aid for the deaf. *Journal of Speech and Hearing Research.* **13**(1), 30-36.

Peterson, J.L. (1980). Computer programs for spelling correction: an experiment in program design. *Lecture Notes in Computer Science* (Goos, G. and Hartmanis J., ed.). **96.**

Pratt, F. (1939). **Secret and Urgent.**

Rhodes, N.W. (1982). An apple talks with the deaf. *Byte.* **7**(1), 366-386.

Riseman, E.M. and Hanson, A.R. (1974). A contextual postprocessing system for error correction using binary n-grams. *IEEE Transactions on Computers.* **C-23**(5), 480-493.

Rosen, M.J. and Goodenough-Trepagnier, C. (1982). Communication Systems for the Non-Vocal Motor Handicapped: Practice and Prospects. *IEEE Engineering in Medicine and Biology.* **1**(4).

Sayre, K.M. (1965). **Recognition: a study in the philosophy of artificial intelligence.**

Schenker, L. (1960). Pushbutton calling with a two-group voice frequency code. *Bell Systems Technical Journal.* **39**(1), 235-255.

Shannon, C.E. (1948). A mathematical theory of communication. *Bell System Technical Journal.* **27**(7), 379-423.

Shannon, C.E. (1948). A mathematical theory of communication. *Bell System Technical Journal.* **27**(10), 623-656.

Shannon, C.E. (1949). Communication theory of secrecy systems. *Bell System Technical Journal.* **28**(10), 656-715.

Shannon, C.E. (1951). Prediction and entropy of printed English. *Bell System Technical Journal.* **30**, 50-64.

Shennib, A. and Kondraske, G. (1984). An improved DTMF telecommunication aid for the deaf. In press.

Shennib, A. and Kondraske, G.V. (1984). An improved telecommunication aid for the deaf. **IEEE Frontiers of Engineering and Computing in Health Care.** 38-40.

Shinghal, R. (1982). An experimental investigation of four text recognition algorithms. *IEEE Transactions on Systems, Man, and Cybernetics.* **SMC-12**(4), 573-577.

Shinghal, R. (1983). A hybrid algorithm for contextual text recognition. *Pattern Recognition.* **16**(2), 261-267.

Sinkov, A. (1976). Elementary Cryptanalysis: a mathematical approach. New Mathematical Library. 22.

Smith, G.M. (1963). Telephone service for the totally deaf. *Volta Review.* **65**, 579-583.

Smith, G.M. (1965). What hath God wrought? *Volta Review.* **67**, 505-507.

Solso, R.L. and King, J.F. (1976). Frequency and versatility of letters in the English language. *Behavior Research Methods & Instrumentation.* **8**(3), 283-286.

Solso, R.L. (1979). Positional frequency and versatility of letters for six-, seven-, and eight-letter English words. *Behavior Research Methods & Instrumentation.* **11**, 355-358.

Solso, R.L., Barbuto, P.F. Jr. and Juel, C.L. (1979). Bigram and Trigram Frequencies and versatilities in the English Language. *Behavior Research Methods & Instrumentation.* **11**(5), 475-484.

Solso, R.L., Juel, C.L. and Rubin, D.C. (1982). The frequency and versatility of initial and terminal letters in English words. *Journal of Verbal Learning and Verbal Behavior.* **21**, 220-235.

Thomas, R.B. and Kassler, M. (1967). Character recognition in context. *Information and Control.* **10**(1), 43-64.

Thorelli, L. (1962). Automatic correction of errors in text. *BIT.* **2**(1), 45-62.

Topper, G.E., Macey, W.H. and Solso, R.L. (1973). Bigram versatility and bigram frequency. *Behavior Research Methods & Instrumentation.* **5**, 51-53.

Wallingford, R. (1974). Let your fingers do the talking. **Proceedings of the Conference on Devices and Systems for the Disabled.** 47-51.

Vanderheiden, G. (1982). **Rehabilitation Aides Resource Book.**

Appendix A. Production Rate Test Materials

Text presented here was spoken verbatim to the subjects.

I am investigating effects of several different methods of entering text on the Touch-Tone keypad. On the sheet to your left is a sample sentence followed by three test sentences. I will read you a set of rules to follow while entering text, and you will enter the test sentence using those rules. During the test sentence I will be watching for, and correcting, your errors. Then, when I say so, enter the three other sentences without pausing.

Enter the text as fast as you can without making errors.

Rule set 1.

Type the key that the letter you want to select is on. Then
type a **1**, **2**, or **3**, depending on whether the letter you want is the
first, second, or third letter on that key.
For example, if you wished to enter an **N**, you would type a
6, then a **2** to signify the second letter on the **6** key.
space is the asterisk key.
Period is two asterisks in a row.

Rule set 2.

Type the key that the letter you want to select is on. Type it
a second time if you wish to select the second letter on that key,
and a third time if you wish to select the third letter on the key.
Type an asterisk to end each letter.
For example, if you wished to enter an **N**, you would type a
6, then another **6**, then an asterisk.
Type two asterisks after a letter if you want to indicate a **space**.
Type three asterisks after a letter for a period.

Rule set 3.

Type the key that the letter you want to select is on.
For example, if you want to enter an **N**, type the **6** key.
space is the asterisk key.
Period is the **0** or operator key.

The Warmup Sentence

I want to go.

The Test Sentences

Are you going to school tomorrow?
She wants to go to a movie.
I will be over in one hour.

Appendix B. Production Rate Data and Calculations

Subject	(order)	Rule Set 1	Rule Set 2	Rule Set 3
P.D.	(1-2-3)	2:55.6	3:26.6	1:15.5
E.K.	(2-1-3)	*3:17.2	3:32.2	1:43.7
O.O.	(1-3-2)	3:16.6	3:57.8	1:37.0
M.0.	(3-1-2)	2:11.9	2:09.5	1:12.0
B.S.	(2-3-1)	2:35.1	3:12.9	1:35.1
J.G.	(3-1-2)	3:00.9	4:04.4	1:41.3
R.K.	(1-2-3)	3:09.8	2:56.9	1:23.9
S.K.	(2-1-3)	3:16.9	3:31.4	1:47.1
M.R.	(1-3-2)	2:15.4	*2:13.6	1:04.9
F.S.	(3-1-2)	2:46.3	2:30.7	1:54.7
K.L.	(2-3-1)	2:51.4	3:27.7	1:29.5
K.C.	(3-2-1)	3:26.3	4:02.1	*2:08.0

Raw Production Rate Test Data

Appendix C. Letter Frequencies of English Text

Letter	Frequency
A	348,411
B	70,714
C	142,336
D	181,054
E	577,583
F	107,219
G	89,499
H	252,191
H	252,191
I	336,166
J	7,296
K	29,838
L	188,261
M	116,574
N	325,652
O	350,121
P	93,040
Q	4,936
R	281,881
S	297,531
T	427,179
U	124,736
V	45,707
W	86,563
X	9,032
Y	78,749
Z	4,316
Space	983,400

Letter Frequencies of English Text.

Appendix D. Digram Frequencies of English Text

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A-	46	7938	16077	15919	314	2580	7393	433	12887	353	4208	36452	10303	71297	187
B-	6003	545	18	41	22104	1	9	9	3705	623	2	8687	122	16	7944
C-	18498	6	2359	14	22238	8	0	21166	8688	0	5997	5323	24	34	27186
D-	5658	101	74	1738	26818	69	1094	151	15960	217	16	1509	599	229	7135
E-	26466	816	15558	45641	15047	5469	4003	892	6121	144	868	19210	13072	50685	2310
F-	6114	13	3	3	8426	5183	6	4	10041	2	5	2348	8	14	18199
G-	5590	29	0	38	13355	56	1104	9780	5358	0	4	2199	252	2178	5471
H-	36624	200	90	63	121957	79	11	39	31931	1	9	515	412	971	19197
I-	8463	3117	23524	11884	12936	7083	9729	33	51	35	2137	16708	12261	87713	25140
J-	916	3	0	10	1550	0	0	0	115	0	0	0	0	0	2082
K-	691	40	6	26	10108	109	75	156	3885	17	19	535	61	2261	353
L-	17871	227	308	10931	30807	1991	205	77	22510	0	1039	22685	1102	195	13909
M-	20065	3422	26	15	29184	152	17	17	11390	0	6	174	3260	354	12620
N-	10892	160	13807	48318	26640	1932	37217	366	12305	374	2223	2675	815	3378	16622
O-	2748	3471	5381	6670	1551	39344	2943	843	3202	222	2937	12768	20158	10963	17
P-	10963	17	9	8	16912	61	16	2975	4894	4	32	9374	615	22	11965
Q-	0	0	0	0	0	0	0	0	0	0	0	11	0	4	0
R-	22760	952	3759	6835	65557	1099	3320	682	24362	6	3461	3567	5654	5982	26375
S-	8635	310	5181	262	31168	521	60	13002	19341	3	1819	2536	2484	495	14045
T-	18525	112	1521	58	41918	304	68	133091	42246	1	15	4272	1078	304	40465
U-	4304	31687	6129	3276	4803	665	5342	46	3751	25	79	12434	4545	15189	396
V-	4183	0	1	5	29756	0	0	0	8788	0	1	5	0	1	2350
W-	18415	64	22	195	13559	46	3	14685	14814	0	57	517	49	3385	8820
X-	856	2	896	0	630	19	0	131	1022	0	0	20	1	3	101
Y-	642	348	316	152	4101	29	73	50	1263	2	39	496	853	416	5859
Z-	772	7	0	0	1	1	6	6	471	0	11	106	3	0	207
SP-	91711	45529	47141	28862	24091	40348	16755	53483	61892	5143	4834	4834	23080	38759	21242

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	PUNCT	
A-	6610	59	38547	34428	52140	4074	7731	2482	731	9427	605	5190	900	
B-	9	0	4047	1377	493	8025	214	10	0	5700	0	893	120	
C-	2	143	5184	685	13972	4601	0	1	0	1162	15	4900	0	
D-	24	67	3585	4451	69	4447	607	293	0	2214	3	103836	4320	
E-	6058	1638	73818	44717	15718	790	9560	4891	6847	6051	177	200915	16800	
F-	0	0	7750	121	3046	3559	0	19	0	286	0	41999	1140	
G-	9	0	7307	1937	645	2756	1	16	0	559	4	30801	5040	
H-	24	18	3150	451	5721	2899	9	198	0	1651	12	25885	6840	
I-	3196	434	11564	40825	40740	400	9159	23	668	7	2288	875	360	
J-	0	0	74	0	0	2411	0	0	0	0	0	11	0	
K-	29	0	100	1709	52	80	1	94	0	379	0	9032	2160	
L-	602	4	422	4669	3858	4601	1120	527	0	17183	17	31146	3780	
M-	7674	0	1404	3441	43	4593	4	24	0	2183	0	16422	2340	
N-	206	187	345	16608	34412	2763	1602	213	98	4139	102	87212	4620	
O-	9	8	16912	61	16	2975	4894	4	32	9374	615	42253	4500	
P-	5130	0	15103	1943	3018	3584	1	41	0	339	0	5931	540	
Q-	0	0	0	0	0	4905	1	2	0	0	0	11	0	
R-	1525	33	4075	15159	12440	4494	2197	428	11	8402	30	58650	5580	
S-	6738	383	122	14014	40310	10387	41	1144	0	1728	18	122649	11400	
T-	88	0	14258	11627	7179	8369	5	2667	0	7102	146	91728	13500	
U-	5248	8	18208	16462	16289	33	88	2	164	213	113	3665	1740	
V-	2	0	21	37	1	82	2	0	0	197	12	241	0	
W-	53	0	1231	1142	195	77	0	3	0	105	1	9041	4260	
X-	2395	7	6	3	1457	128	1	13	4	84	0	1252	120	
Y-	681	0	374	3290	791	49	7	235	0	3	56	58623	6600	
Z-	0	0	4	2	1	29	8	5	0	107	338	282	0	
SP-	71249	38525	1913	25769	67966	58974	11314	6395	60167	16	8074	211	0	

Appendix E. Trigram Frequencies of English Text

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A-A-	1	1	15	1	1	1	0	1	0	0	3	0	4	0	0
A-B-	157	136	1	14	179	0	0	2	583	5	0	3513	0	6	2544
A-C-	215	5	1343	0	2839	0	0	2763	383	0	2589	38	1	7	95
A-D-	294	9	37	859	2731	20	21	40	1328	207	0	162	488	32	355
A-E-	5	0	3	1	0	13	16	0	0	0	1	44	1	0	21
A-F-	27	0	0	0	194	668	4	1	10	0	5	15	1	0	12
A-G-	1390	0	0	3	3470	0	303	11	445	0	0	15	50	264	566
A-H-	50	0	2	0	0	114	0	0	11	1	1	14	20	9	36
A-I-	1	0	52	2538	18	0	277	0	22	3	6	1559	397	5529	0
A-J-	8	0	0	10	16	0	0	0	2	0	0	0	0	0	311
A-K-	45	0	0	16	2895	57	0	5	654	0	2	9	1	52	18
A-L-	572	79	153	125	1343	436	45	12	2318	0	758	10153	657	29	774
A-M-	473	363	5	2	4128	6	0	0	1020	0	0	17	224	70	798
A-N-	945	10	3505	32000	987	20	2332	37	1748	4	830	54	2	1428	1019
A-O-	0	0	0	2	0	0	0	0	9	0	0	0	2	0	0
A-P-	541	2	0	4	877	2	2	432	363	0	5	40	6	3	250
A-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A-R-	1305	318	829	3579	7668	85	1293	6	2290	1	1379	1570	1134	704	787
A-S-	134	12	148	0	2116	0	2	827	1261	0	810	38	71	2	707
A-T-	364	3	626	0	10595	88	0	2144	12155	0	10	191	238	34	973
A-U-	1	11	91	277	15	11	565	1	1	3	11	232	13	157	0
A-V-	481	0	0	0	5717	0	0	0	905	0	0	4	0	0	427
A-W-	634	8	3	9	46	36	0	8	78	0	58	95	12	153	13
A-X-	28	0	0	0	105	0	0	0	203	0	0	6	0	2	33
A-Y-	57	181	6	8	549	10	4	6	342	0	0	47	147	72	72
A-Z-	57	0	0	0	133	0	0	0	173	0	0	2	0	0	30
A-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
A-A	0	0	15	1	0	0	0	1	0	0	0	0	0	
A-B	0	0	198	349	0	103	0	0	0	35	0	58	0	
A-C	0	143	482	15	4437	178	0	1	0	223	0	69	0	
A-D	1	66	60	282	7	229	600	41	0	586	1	7463	0	
A-E	1	0	77	47	5	17	1	0	0	0	0	61	0	
A-F	0	0	151	1	1452	0	0	0	0	1	0	38	0	
A-G	6	0	459	36	0	215	0	3	0	1	0	156	0	
A-H	0	0	4	6	2	1	1	1	0	0	11	149	0	
A-I	2	0	1537	362	536	1	12	7	1	0	1	26	0	
A-J	0	0	0	0	0	4	0	0	0	0	0	2	0	
A-K	0	0	3	66	9	9	0	9	0	14	0	344	0	
A-L	63	0	328	2099	1083	585	112	470	0	300	4	13954	0	
A-M	1061	0	12	423	4	114	0	2	0	31	0	1550	0	
A-N	14	22	0	2394	4509	261	39	35	86	3410	18	13810	0	
A-O	0	0	37	82	13	6	0	0	1	0	0	35	0	
A-P	2898	0	87	435	373	10	0	0	0	20	0	260	0	
A-Q	0	0	0	0	0	53	1	2	0	0	0	3	0	
A-R	237	13	1658	1997	4588	13	180	29	11	2182	13	4643	0	
A-S	256	9	2	3111	3435	526	0	2	0	197	0	20762	0	
A-T	0	0	268	415	2518	1450	0	4	0	41	14	20009	0	
A-U	2	2	157	1430	944	1	3	0	30	0	1	115	0	
A-V	0	0	8	1	0	9	1	0	0	155	0	20	0	
A-W	2	0	46	145	17	0	0	1	0	67	0	853	0	
A-X	33	0	0	3	12	0	0	12	0	5	0	289	0	
A-Y	0	0	22	1469	56	2	0	10	0	0	0	6367	0	
A-Z	0	0	0	0	0	3	0	1	0	50	154	2	0	
A-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
A-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
B-A-	0	455	1308	201	5	12	87	8	34	0	70	681	25	758	1
B-B-	16	0	0	0	153	0	0	0	121	0	0	88	0	0	50
B-C-	2	0	0	0	0	0	0	0	0	0	0	0	0	0	16
B-D-	2	0	0	0	0	0	0	0	12	0	0	0	0	0	12
B-E-	767	1	1710	647	2643	1030	735	430	771	0	4	1117	4	436	14
B-F-	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
B-G-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B-H-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
B-I-	61	93	53	92	73	5	474	0	0	1	1	1209	4	512	81
B-J-	0	0	0	0	619	0	0	0	0	0	0	0	0	0	0
B-K-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
B-L-	555	0	0	3	5000	0	0	0	1507	0	0	0	0	0	505
B-M-	55	0	0	0	8	0	0	0	57	2	0	0	0	0	0
B-N-	1	0	0	0	3	0	0	0	0	0	0	0	0	0	11
B-O-	616	95	8	679	5	2	24	19	57	0	0	301	120	341	486
B-P-	2	0	0	0	3	0	0	0	0	0	0	0	0	0	1
B-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B-R-	708	0	0	0	585	0	0	0	1205	0	0	0	0	0	1263
B-S-	0	1	62	0	419	0	0	0	47	0	5	0	0	0	139
B-T-	181	0	0	0	47	24	0	0	9	0	0	50	0	0	0
B-U-	1	33	108	158	6	63	34	2	526	0	4	278	56	174	5
B-V-	0	0	0	0	7	0	0	0	207	0	0	0	0	0	0
B-W-	8	0	0	0	1	0	0	0	0	0	0	0	0	0	1
B-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B-Y-	2	0	0	0	10	0	2	2	1	0	0	13	0	0	0
B-Z-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
B-A	44	0	726	917	463	21	3	9	0	90	8	77	0	
B-B	0	0	3	12	0	2	0	0	0	69	0	31	0	
B-C	0	0	0	0	0	0	0	0	0	0	0	0	0	
B-D	0	0	1	0	0	13	0	0	0	0	0	1	0	
B-E	7	10	2861	663	1292	1	36	20	1	218	3	6683	0	
B-F	0	0	0	0	0	0	0	0	0	0	0	0	0	
B-G	0	0	9	0	0	0	0	0	0	0	0	0	0	
B-H	0	0	0	0	0	1	0	0	0	0	0	0	0	
B-I	4	2	248	85	642	0	16	9	1	0	12	27	0	
B-J	0	0	0	0	0	4	0	0	0	0	0	0	0	
B-K	0	0	0	0	0	0	0	0	0	0	0	0	0	
B-L	0	0	0	0	0	266	4	0	0	247	0	0	0	
B-M	0	0	0	0	0	0	0	0	0	0	0	0	0	
B-N	0	0	0	0	0	0	0	0	0	0	0	1	0	
B-O	4	0	950	120	997	2140	302	92	112	457	0	17	0	
B-P	0	0	3	0	0	0	0	0	0	0	0	0	0	
B-Q	0	0	0	0	0	0	0	0	0	0	0	0	0	
B-R	0	0	0	0	0	262	0	0	0	23	0	1	0	
B-S	11	0	0	0	371	30	0	0	0	12	0	280	0	
B-T	0	0	23	28	0	0	0	0	0	4	0	127	0	
B-U	0	0	780	703	4942	0	0	0	3	119	23	5	0	
B-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
B-W	0	0	0	0	0	0	0	0	0	0	0	0	0	
B-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
B-Y	9	0	30	10	12	0	0	1	0	0	7	5601	0	
B-Z	0	0	0	0	0	0	0	0	0	0	0	0	0	
B-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
B-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
C-A-	0	187	38	293	9	53	116	2	18	0	18	4601	1264	3655	2
C-B-	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0
C-C-	212	0	0	0	809	0	0	10	72	0	0	33	0	0	639
C-D-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14
C-E-	148	3	7	1513	213	71	1	2	469	0	0	573	273	2215	3
C-F-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-G-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-H-	3108	14	2	13	2951	7	8	10	2509	0	3	102	95	389	1549
C-I-	2050	19	17	468	1281	316	50	0	0	0	0	357	48	626	359
C-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-K-	106	25	0	7	1048	8	74	39	109	2	0	253	32	116	48
C-L-	1332	0	0	0	1553	0	0	0	424	0	0	0	0	0	952
C-M-	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0
C-N-	0	0	0	0	11	0	0	0	23	0	0	0	0	0	0
C-O-	363	38	121	93	47	95	232	63	85	0	5	1677	6869	8869	496
C-P-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-R-	757	0	0	0	2065	0	0	0	1061	0	0	0	0	0	862
C-S-	0	0	0	0	0	0	0	0	2	0	0	0	0	0	4
C-T-	268	0	0	0	1493	8	0	0	5161	0	0	360	22	7	1065
C-U-	10	122	0	8	45	8	0	0	40	0	0	1456	276	22	19
C-V-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-W-	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-Y-	5	0	68	0	0	0	1	0	1	0	0	37	0	22	1
C-Z-	6	0	0	0	6	0	0	0	0	0	0	0	0	0	0
C-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
C-A	814	0	1986	1108	2644	1865	87	2	28	0	0	308	0	
C-B	0	0	0	0	0	0	0	0	0	0	0	0	0	
C-C	0	0	25	0	0	557	0	0	0	0	0	2	0	
C-D	0	0	0	0	0	0	0	0	0	0	0	0	0	
C-E	883	0	1675	3242	64	2	2	4	0	14	3	10858	0	
C-F	0	0	0	0	0	0	0	0	0	0	0	8	0	
C-G	0	0	0	0	0	0	0	0	0	0	0	0	0	
C-H	0	0	473	16	34	608	1	38	0	44	0	9192	0	
C-I	633	0	314	666	1284	25	213	0	0	0	33	9	0	
C-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
C-K	25	0	8	448	30	19	1	49	0	81	0	3264	0	
C-L	0	0	0	0	0	1030	0	0	0	32	0	0	0	
C-M	0	0	0	0	0	0	0	0	0	0	0	21	0	
C-N	0	0	0	0	0	0	0	0	0	0	0	0	0	
C-O	277	2	1737	564	192	4361	569	153	10	9	3	256	0	
C-P	0	0	0	2	0	0	0	0	0	0	0	0	0	
C-Q	0	0	0	0	0	143	0	0	0	0	0	0	0	
C-R	0	0	0	0	0	292	0	0	0	147	0	0	0	
C-S	0	0	0	0	10	0	0	0	0	0	0	669	0	
C-T	0	0	451	984	0	1067	0	1	0	2	0	3063	0	
C-U	221	0	1188	626	535	23	0	0	0	0	0	2	0	
C-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
C-W	0	0	0	0	0	0	0	0	0	0	0	0	0	
C-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
C-Y	8	0	7	3	10	0	0	0	0	0	0	999	0	
C-Z	0	0	0	0	0	0	0	0	0	1	0	2	0	
C-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
C-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
D-A-	0	81	30	26	6	1	37	7	176	0	8	189	373	686	0
D-B-	13	0	0	0	17	0	0	0	9	0	0	3	0	0	16
D-C-	42	0	0	0	0	0	0	6	0	0	0	3	0	0	19
D-D-	32	0	0	0	599	0	0	23	474	0	0	232	0	0	12
D-E-	1407	145	1074	2566	464	689	169	6	14	11	1	844	781	3334	61
D-F-	17	0	0	0	0	0	0	0	5	0	0	3	0	0	10
D-G-	4	0	0	0	928	0	0	2	40	0	3	4	93	0	0
D-H-	22	0	0	0	52	0	0	0	16	0	0	0	0	0	61
D-I-	934	47	1195	1173	1003	1049	157	0	5	1	0	196	169	3266	299
D-J-	19	0	0	0	12	0	0	0	0	0	0	0	0	0	25
D-K-	0	0	0	0	10	0	0	0	4	0	0	0	0	0	2
D-L-	17	0	0	0	635	0	0	0	138	0	0	0	0	0	31
D-M-	57	0	0	0	43	0	0	0	472	0	0	0	0	0	25
D-N-	17	0	0	0	172	0	0	0	29	0	0	0	0	0	6
D-O-	11	6	301	24	494	2	191	1	179	0	0	298	604	697	459
D-P-	5	0	0	0	1	0	0	0	0	0	0	0	0	0	17
D-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D-R-	636	0	0	0	1209	0	0	0	698	0	0	0	0	0	488
D-S-	2	0	30	0	2	0	0	67	17	0	1	5	10	0	131
D-T-	3	0	0	0	0	0	0	34	4	0	0	1	0	0	0
D-U-	546	22	1707	3	189	9	25	3	5	0	13	262	60	52	4
D-V-	265	0	0	0	151	0	0	0	161	0	0	0	0	0	30
D-W-	127	0	0	0	62	0	0	1	70	0	0	0	0	0	24
D-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D-Y-	6	9	0	0	9	0	2	0	81	0	10	18	4	61	0
D-Z-	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
D-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY
D-A	63	0	711	49	659	58	125	33	0	2018	20	262	0
D-B	0	0	1	0	0	32	0	0	0	10	0	0	0
D-C	0	0	1	0	0	3	0	0	0	0	0	0	0
D-D	0	0	128	24	0	1	0	0	0	72	0	141	0
D-E	990	126	5347	2392	782	16	1063	61	97	2	14	4362	0
D-F	0	0	1	0	0	33	0	0	0	0	0	0	0
D-G	0	0	4	0	0	6	0	4	0	3	0	3	0
D-H	0	0	0	0	0	0	0	0	0	0	0	0	0
D-I	108	0	803	3036	1461	116	829	1	21	0	25	66	0
D-J	0	0	0	0	0	161	0	0	0	0	0	0	0
D-K	0	0	0	0	0	0	0	0	0	0	0	0	0
D-L	0	0	0	0	0	8	0	0	0	677	0	3	0
D-M	0	0	0	0	0	2	0	0	0	0	0	0	0
D-N	0	0	0	0	0	0	0	0	0	0	0	5	0
D-O	94	0	209	56	57	414	14	1403	52	4	73	1492	0
D-P	0	0	1	0	0	67	0	0	0	0	0	0	0
D-Q	0	67	0	0	0	0	0	0	0	0	0	0	0
D-R	0	0	0	5	0	222	0	0	0	134	0	193	0
D-S	10	0	0	0	59	3	6	2	0	0	0	4106	0
D-T	0	0	0	0	0	0	0	0	0	0	0	27	0
D-U	32	1	827	491	117	0	1	0	0	1	0	77	0
D-V	0	0	0	0	0	0	0	0	0	0	0	0	0
D-W	0	0	5	0	0	0	0	0	0	3	0	1	0
D-X	0	0	0	0	0	0	0	0	0	0	0	0	0
D-Y	0	0	0	34	14	0	0	8	0	0	0	1958	0
D-Z	0	0	0	0	0	0	0	0	0	1	0	0	0
D-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0
D-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
E-A-	0	80	2193	3136	0	79	212	29	0	0	767	2004	578	1499	0
E-B-	136	11	0	0	116	0	0	0	11	0	0	18	0	0	113
E-C-	1439	0	55	14	1589	0	0	676	1599	0	421	244	0	0	2315
E-D-	62	11	17	146	807	7	367	16	1447	0	0	204	10	64	204
E-E-	18	26	133	1880	4	59	25	9	113	0	767	802	1041	4188	0
E-F-	27	0	1	3	732	1053	0	0	441	0	0	241	0	0	1332
E-G-	1064	8	0	0	505	4	98	4	893	0	0	59	23	18	198
E-H-	202	0	0	0	134	1	0	2	351	0	0	15	3	1	147
E-I-	5	2	36	35	2	2	878	0	5	0	7	102	27	1131	0
E-J-	11	0	0	0	79	0	0	0	0	0	0	0	0	0	21
E-K-	3	7	0	2	71	0	0	2	52	0	2	29	0	3	5
E-L-	1064	25	98	964	2116	1866	40	10	2087	0	22	3256	93	1	1237
E-M-	1169	1367	2	1	3066	0	0	0	663	0	4	17	71	69	1041
E-N-	638	31	4264	3471	2688	73	1230	91	963	179	25	284	13	470	740
E-O-	0	0	34	20	1	27	54	0	4	0	0	163	37	211	0
E-P-	846	0	2	0	638	0	0	250	301	0	1	340	5	2	576
E-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E-R-	4215	192	857	203	12575	728	627	371	4432	5	95	524	1544	2716	584
E-S-	75	32	675	96	3351	2	1	245	1681	0	76	19	103	16	340
E-T-	881	19	272	0	1816	7	0	1784	1439	0	2	69	14	17	178
E-U-	0	7	3	33	3	0	42	0	2	0	3	3	62	27	0
E-V-	374	0	1	0	7521	0	0	0	1147	0	0	0	0	0	366
E-W-	189	22	16	16	243	5	0	236	269	0	0	34	11	8	48
E-X-	813	1	885	0	318	0	0	128	514	0	0	8	0	0	30
E-Y-	26	13	1	8	637	0	0	4	18	0	1	8	18	25	185
E-Z-	3	0	0	0	75	0	0	0	23	0	0	0	0	0	5
E-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
E-A	248	3	6423	3778	3664	350	680	14	0	0	0	726	0	
E-B	0	0	174	16	35	89	0	1	0	63	0	33	0	
E-C	0	0	501	11	6065	532	0	0	0	14	0	83	0	
E-D	1	0	93	367	6	881	2	66	0	277	1	32000	0	
E-E	909	0	547	524	1364	0	23	15	2	0	81	2517	0	
E-F	0	0	104	33	512	558	0	0	0	11	0	421	0	
E-G	0	0	12	75	0	285	0	0	0	45	0	112	0	
E-H	0	0	10	0	0	5	0	0	0	3	0	18	0	
E-I	18	0	2721	126	503	0	419	0	0	0	43	63	0	
E-J	0	0	0	0	0	28	0	0	0	0	0	5	0	
E-K	0	0	0	197	6	0	0	1	0	0	0	488	0	
E-L	596	0	6	787	659	73	462	0	0	2889	5	1854	0	
E-M	1292	0	1	1112	0	29	0	0	0	139	0	3029	0	
E-N	7	3	193	2325	17817	246	153	37	0	70	49	14625	0	
E-O	910	0	475	6	24	204	90	17	0	0	0	33	0	
E-P	81	0	467	191	1372	239	0	4	8	1	0	735	0	
E-Q	0	4	0	0	0	1631	0	0	0	0	0	735	0	
E-R	353	1	818	7987	2137	88	1713	223	0	2333	5	28492	0	
E-S	1296	33	0	7116	6537	704	7	7	0	11	1	22293	0	
E-T	4	0	446	716	1678	1678	1	777	0	470	7	4712	0	
E-U	24	0	345	46	134	0	29	0	12	0	0	15	0	
E-V	0	0	4	6	0	11	1	0	0	12	1	116	0	
E-W	46	0	21	387	19	3	0	0	0	2	0	3319	0	
E-X	2362	7	6	0	1297	97	0	0	0	3	0	378	0	
E-Y	0	0	1	105	7	0	1	6	0	0	0	4987	0	
E-Z	0	0	2	0	0	5	7	0	0	1	9	47	0	
E-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
E-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
F-A-	0	93	2056	29	2	4	6	6	730	0	13	391	626	188	0
F-B-	11	0	0	0	1	0	0	0	0	0	0	0	0	0	1
F-C-	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
F-D-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
F-E-	485	72	867	382	1092	1	7	0	12	0	1	685	83	427	0
F-F-	146	2	0	0	2007	0	0	2	1389	0	0	90	5	11	362
F-G-	1	0	0	0	0	0	0	4	0	0	0	0	0	0	0
F-H-	2	0	0	0	1	0	0	0	0	0	0	0	0	0	1
F-I-	51	73	2246	134	966	185	678	0	0	0	3	649	0	1957	5
F-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
F-K-	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F-L-	411	0	0	0	575	0	0	0	232	0	0	0	0	1	578
F-M-	6	0	0	0	0	0	0	0	1	0	0	0	0	0	1
F-N-	1	0	0	0	12	0	0	0	1	0	0	0	0	0	0
F-O-	80	0	86	2	17	0	58	5	26	0	0	789	0	32	438
F-P-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F-R-	783	0	0	0	1232	0	0	0	771	0	0	0	0	0	4856
F-S-	1	0	0	0	10	0	0	3	0	0	4	0	0	0	1
F-T-	8	0	0	0	1824	1	0	46	54	0	0	47	0	8	14
F-U-	0	0	24	14	26	0	49	2	0	3	0	1727	31	611	0
F-V-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F-W-	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F-Y-	0	0	1	0	0	0	0	0	73	0	0	0	0	0	1
F-Z-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
F-A	0	1	867	335	410	78	230	9	4	25	2	9	0	
F-B	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-C	0	0	0	0	0	0	0	0	0	0	0	1	0	
F-D	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-E	0	0	1933	404	117	20	28	634	0	1	0	1175	0	
F-F	0	0	23	59	1	45	0	0	0	12	0	1029	0	
F-G	0	0	0	0	0	0	0	0	0	0	0	1	0	
F-H	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-I	0	1	2009	289	360	0	288	0	143	0	3	1	0	
F-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-K	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-L	0	0	0	0	0	791	0	0	0	160	0	0	0	
F-M	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-N	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-O	1	0	15353	46	0	1241	0	3	17	5	0	0	0	
F-P	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-Q	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-R	0	0	0	0	0	105	0	0	0	3	0	0	0	
F-S	7	0	0	0	2	0	0	0	0	0	0	93	0	
F-T	0	0	1	62	0	1	0	2	0	83	0	895	0	
F-U	0	0	507	304	248	1	0	0	0	0	11	1	0	
F-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-W	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-Y	0	0	0	0	0	0	0	0	0	0	0	211	0	
F-Z	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
F-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
G-A-	1	31	6	28	9	0	165	0	1427	0	1	293	243	1111	0
G-B-	3	0	0	0	3	0	0	0	0	0	0	0	0	0	15
G-C-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G-D-	3	0	0	0	3	0	0	0	0	0	0	0	0	0	31
G-E-	120	7	3	1017	36	1	1	7	6	0	0	261	246	1984	333
G-F-	0	0	0	0	3	0	0	0	3	0	0	0	0	0	19
F-G-	23	0	0	0	606	0	0	1	144	0	0	158	0	0	4
G-H-	55	163	3	0	392	5	2	0	49	0	2	182	25	16	188
G-I-	184	104	477	45	110	74	21	0	1	0	0	77	60	1789	508
G-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G-K-	0	0	0	0	0	0	0	0	3	0	0	0	0	0	1
G-L-	561	0	0	0	586	0	0	0	405	0	0	0	0	0	216
G-M-	41	0	0	0	180	0	0	0	3	0	0	0	0	0	0
G-N-	230	1	0	0	426	0	0	0	595	0	0	0	87	0	152
G-O-	114	6	5	383	102	0	15	0	458	0	0	185	19	453	934
G-P-	0	0	0	0	1	0	0	0	5	0	0	2	0	0	1
G-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G-R-	2413	0	0	0	2659	0	0	0	363	0	0	0	0	0	1724
G-S-	1	2	0	1	0	0	0	5	17	0	8	2	0	0	2
G-T-	0	0	0	0	1	0	0	351	5	0	0	0	0	0	287
G-U-	329	7	0	0	610	1	4	1	461	0	0	315	105	274	28
G-V-	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
G-W-	4	0	0	0	1	0	0	0	5	0	0	0	0	0	5
G-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G-Y-	0	0	0	0	0	0	0	0	0	0	0	0	41	6	0
G-Z-	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
G-A	31	0	685	193	785	53	306	4	0	37	106	75	0	
G-B	0	0	1	0	0	0	0	0	0	7	0	0	0	
G-C	0	0	0	0	0	0	0	0	0	0	0	0	0	
G-D	0	0	0	0	0	0	0	0	0	0	0	1	0	
G-E	3	0	1789	1408	1596	4	4	9	0	18	0	4502	0	
G-F	0	0	1	0	0	30	0	0	0	0	0	0	0	
G-G	0	0	59	43	0	0	0	0	0	25	0	41	0	
G-H	3	0	6	31	5670	7	0	58	0	1	0	2922	0	
G-I	1	0	380	352	68	10	1062	0	0	0	25	10	0	
G-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
G-K	0	0	0	0	0	0	0	0	0	0	0	0	0	
G-L	0	0	0	0	0	51	0	0	0	380	0	0	0	
G-M	0	0	0	2	0	1	0	0	0	0	0	25	0	
G-N	3	0	0	122	26	19	0	0	0	0	0	515	0	
G-O	0	0	242	61	639	55	670	24	0	1	0	1105	0	
G-P	0	0	0	0	0	0	0	0	0	0	0	0	0	
G-Q	0	0	0	0	0	0	0	0	0	0	0	0	0	
G-R	0	0	0	0	0	78	0	0	0	68	1	1	0	
G-S	0	0	0	0	65	1	0	6	0	0	0	1827	0	
G-T	0	0	1	0	0	0	0	0	0	0	0	0	0	
G-U	0	0	405	120	23	0	0	0	0	71	2	0	0	
G-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
G-W	0	0	1	0	0	0	0	0	0	0	0	0	0	
G-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
G-Y	35	0	36	0	0	0	0	0	0	0	0	441	0	
G-Z	0	0	0	0	0	0	0	0	0	0	0	0	0	
G-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
G-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
H-A-	5	173	37	5271	58	42	44	8	455	4	113	1086	427	4565	22
H-B-	8	0	0	0	2	0	0	0	8	0	0	1	0	0	173
H-C-	3	0	0	0	2	0	0	70	0	0	0	0	0	0	14
H-D-	20	0	0	0	0	0	0	0	1	0	0	0	0	0	10
H-E-	2643	19	174	1848	403	24	6	3	2841	0	4	1466	2443	4471	314
H-F-	4	0	0	0	0	0	0	0	10	0	0	0	0	0	5
H-G-	2	0	0	0	2	0	0	0	0	0	0	0	0	0	7
H-H-	0	0	0	0	17	0	0	2	0	0	0	0	0	0	13
H-I-	155	163	4206	98	519	103	956	1	3	6	19	2049	3277	5126	184
H-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H-K-	3	0	0	0	2	0	0	0	1	0	0	0	0	0	2
H-L-	12	9	0	0	115	0	0	0	33	0	0	0	0	0	80
H-M-	104	0	0	0	181	0	0	0	20	0	0	0	1	1	18
H-N-	12	0	0	0	47	1	1	0	327	0	0	0	0	56	63
H-O-	54	25	99	530	118	6	41	10	147	1	25	1622	1069	647	1220
H-P-	6	0	0	0	0	0	0	0	8	0	0	6	0	0	3
H-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H-R-	97	0	0	0	904	0	0	0	477	0	0	0	1	0	1490
H-S-	4	0	1	0	2	0	0	3	1	0	0	1	0	0	7
H-T-	10	0	6	1	463	87	0	30	123	0	0	153	12	37	9
H-U-	7	28	34	91	5	20	114	9	3	0	0	33	565	481	0
H-V-	0	0	0	0	1	0	0	0	8	0	0	0	0	0	0
H-W-	108	0	0	0	60	0	0	10	12	0	0	0	0	0	7
H-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H-Y-	13	1	0	87	1	5	3	0	0	0	0	40	41	7	0
H-Z-	0	0	0	0	11	0	0	0	1	0	0	0	0	0	0
H-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY
H-A	1093	1	2454	2846	13066	116	4430	105	3	101	51	48	0
H-B	0	0	7	0	0	0	0	0	0	1	0	0	0
H-C	0	0	0	0	0	1	0	0	0	0	0	0	0
H-D	0	0	31	0	0	0	0	0	0	0	0	1	0
H-E	14	5	15467	2926	515	16	101	48	11	3658	10	16991	0
H-F	0	0	0	0	0	59	0	0	0	0	0	1	0
H-G	0	0	0	0	0	0	0	0	0	0	0	0	0
H-H	0	0	0	0	0	4	0	0	0	0	0	3	0
H-I	986	2	463	12821	705	1	34	0	0	0	28	44	0
H-J	0	0	0	0	0	1	0	0	0	0	0	0	0
H-K	0	0	0	0	0	0	0	0	0	0	0	1	0
H-L	0	0	0	0	0	1	0	0	0	237	0	28	0
H-M	1	0	0	26	0	2	0	0	0	0	0	58	0
H-N	0	0	0	74	0	1	0	0	0	0	0	389	0
H-O	620	0	1403	1557	584	4659	53	2407	1	7	4	2288	0
H-P	0	0	0	0	0	1	0	0	0	0	0	0	0
H-Q	0	0	0	0	0	18	0	0	0	0	0	0	0
H-R	0	0	1	0	0	158	0	0	0	4	0	18	0
H-S	0	0	0	0	14	0	0	0	0	0	0	418	0
H-T	0	0	1	344	13	5	0	8	0	60	0	4359	0
H-U	4	0	740	610	105	0	0	0	6	14	2	28	0
H-V	0	0	0	0	0	0	0	0	0	0	0	0	0
H-W	0	0	0	0	0	1	0	0	0	0	0	0	0
H-X	0	0	0	0	0	0	0	0	0	0	0	0	0
H-Y	121	0	87	323	50	0	0	0	0	0	0	872	0
H-Z	0	0	0	0	0	0	0	0	0	0	0	0	0
H-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0
H-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
I-A-	0	216	39	30	5	0	233	11	7	0	5	3282	339	1702	3
I-B-	41	56	1	0	628	0	0	0	531	0	0	1164	0	1	31
I-C-	5525	0	9	0	2982	0	0	4015	1562	0	1292	301	0	20	122
I-D-	403	3	0	238	5415	0	251	0	391	3	0	140	2	57	79
I-E-	0	8	267	1688	0	510	23	2	0	0	11	603	6	2222	0
I-F-	8	0	0	0	1128	1186	0	0	1273	0	0	120	0	1	173
I-G-	358	12	0	0	263	0	148	5428	655	0	0	10	39	1611	102
I-H-	3	0	0	0	0	0	0	0	12	0	0	0	1	1	15
I-I-	6	0	0	0	0	0	0	1	0	1	0	0	0	9	0
I-J-	10	3	0	0	3	0	0	0	2	0	0	0	0	0	2
I-K-	12	0	0	0	1976	0	0	1	76	0	1	1	0	0	22
I-L-	683	43	11	1248	2349	8	49	37	2031	0	78	6093	175	3	291
I-M-	1270	158	8	0	3049	1	0	0	968	0	0	33	492	11	142
I-N-	2314	34	3157	4249	5264	918	29764	147	2097	115	1079	249	56	673	328
I-O-	13	1	27	382	6	0	41	0	2	0	0	324	27	21626	0
I-P-	356	4	2	0	128	2	1	39	63	4	0	392	195	0	25
I-Q-	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0
I-R-	260	18	403	339	2598	15	127	0	590	0	38	501	430	10	197
I-S-	377	33	1108	94	1738	196	43	2243	1499	3	179	445	824	4	440
I-T-	1620	1	432	0	3397	18	2	9089	4716	0	1	204	43	84	441
I-U-	1	1	2	3	0	0	0	0	0	0	1	2	303	1	0
I-V-	599	0	0	0	6599	0	0	0	1835	0	0	0	0	1	98
I-W-	19	0	0	0	2	0	0	0	1	0	0	0	0	0	1
I-X-	1	0	0	0	148	0	0	0	30	0	0	0	0	0	34
I-Y-	2	0	0	0	2	0	0	0	0	0	0	0	0	0	2
I-Z-	587	0	0	0	1884	0	0	0	158	0	0	0	0	0	62
I-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
I-A	15	0	216	115	1239	1	1	2	0	0	25	977	0	
I-B	0	0	174	25	0	447	0	0	0	7	0	11	0	
I-C	0	0	99	634	1326	830	0	0	0	257	2	4548	0	
I-D	2	0	1	208	24	378	0	30	0	12	0	4247	0	
I-E	20	0	721	3833	927	71	692	475	0	0	20	837	0	
I-F	0	0	0	7	538	189	0	0	0	231	0	2229	0	
I-G	1	0	95	42	0	454	0	0	0	3	4	504	0	
I-H	0	0	0	0	0	0	0	0	0	0	0	1	0	
I-I	0	0	0	4	0	0	0	0	0	0	0	30	0	
I-J	0	0	0	0	0	11	0	0	0	0	0	4	0	
I-K	0	0	0	14	0	4	0	0	0	0	0	30	0	
I-L	15	2	77	341	304	121	36	26	0	1059	1	1627	0	
I-M	2052	0	0	754	1	289	0	0	0	0	0	3033	0	
I-N	29	115	4	3208	6309	847	856	15	2	104	8	25770	0	
I-O	20	0	544	82	69	1631	6	5	3	0	1	330	0	
I-P	398	0	11	332	185	51	0	3	0	4	0	1001	0	
I-Q	0	0	0	0	0	423	0	0	0	0	0	0	0	
I-R	50	0	227	1655	454	19	12	23	0	60	1	3537	0	
I-S	445	14	72	1577	5733	102	6	1	0	13	0	23636	0	
I-T	0	0	94	2718	1939	1210	0	3	0	3983	92	10653	0	
I-U	0	0	6	79	0	0	1	0	0	0	0	0	0	
I-V	0	0	4	2	0	5	0	0	0	14	0	2	0	
I-W	0	0	0	0	0	0	0	0	0	0	0	0	0	
I-X	0	0	0	0	141	0	0	0	0	0	0	314	0	
I-Y	0	0	0	0	0	1	0	0	0	0	0	0	0	
I-Z	0	0	0	0	0	9	1	0	0	0	67	20	0	
I-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
I-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
J-A-	0	7	246	3	0	0	17	6	27	0	9	1	138	150	0
J-B-	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0
J-C-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-D-	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-E-	35	1	944	8	19	33	0	1	0	3	0	12	1	44	9
J-F-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-G-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-H-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-I-	0	1	0	1	0	2	10	0	0	0	0	16	55	6	0
J-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-K-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-L-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-M-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-N-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-O-	16	308	14	1	69	0	1	482	240	0	39	33	0	84	0
J-P-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-R-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-S-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-T-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-U-	33	14	1	298	8	0	46	1	20	1	2	120	80	296	0
J-V-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-W-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-Y-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-Z-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
J-A	95	0	37	18	0	2	5	28	0	19	101	7	0	
J-B	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-C	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-D	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-E	0	0	90	138	36	3	0	169	0	0	0	4	0	
J-F	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-G	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-H	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-I	0	0	0	7	4	0	1	0	0	0	0	12	0	
J-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-K	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-L	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-M	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-N	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-O	1	0	325	74	3	136	3	6	0	232	0	15	0	
J-P	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-Q	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-R	0	0	0	0	0	0	0	0	0	0	0	74	0	
J-S	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-T	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-U	9	0	197	1255	3	0	18	0	5	0	0	4	0	
J-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-W	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-Y	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-Z	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
J-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
K-A-	0	112	0	36	0	5	54	9	3	1	2	18	11	69	0
K-B-	1	0	0	0	15	0	0	0	1	0	0	0	0	0	22
K-C-	3	0	0	0	0	0	0	1	0	0	0	0	0	0	0
K-D-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19
K-E-	12	2	0	2142	492	20	8	13	24	0	1	241	13	923	9
K-F-	58	0	0	0	2	0	0	0	3	0	0	0	0	0	6
K-G-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
K-H-	0	0	0	0	16	0	0	0	0	0	0	0	1	0	44
K-I-	10	1	52	129	76	13	0	0	11	0	5	393	29	2750	2
K-J-	2	0	0	0	0	0	0	0	0	0	0	0	0	0	15
K-K-	2	0	0	0	11	0	0	0	4	0	0	0	0	0	1
K-L-	48	188	0	0	0	0	0	0	83	0	0	0	0	0	34
K-M-	42	0	0	0	15	0	0	0	1	0	0	0	0	0	2
K-N-	22	0	0	0	686	0	0	0	134	0	0	0	0	0	1406
K-O-	1	3	3	13	7	45	1	15	3	1	5	31	6	56	4
K-P-	3	0	0	0	0	0	0	0	20	0	0	0	0	0	6
K-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K-R-	19	0	0	0	14	0	0	0	24	0	0	0	0	0	15
K-S-	1	4	3	0	4	0	9	42	6	0	8	10	13	0	55
K-T-	28	0	0	0	0	0	0	7	8	0	0	0	0	0	6
K-U-	0	1	0	0	0	0	0	1	0	0	1	14	1	4	0
K-V-	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
K-W-	59	0	0	0	4	0	0	4	3	0	0	0	0	0	22
K-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K-Y-	6	1	0	0	1	0	0	0	0	2	0	12	0	3	26
K-Z-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
K-A	10	0	70	20	112	4	1	4	1	55	3	91	0	
K-B	0	0	1	0	0	0	0	0	0	0	0	0	0	
K-C	0	0	0	0	0	0	0	0	0	0	0	2	0	
K-D	0	0	3	0	0	0	0	0	0	0	0	0	0	
K-E	204	0	734	561	650	1	0	26	0	229	1	3802	0	
K-F	0	0	0	0	0	40	0	0	0	0	0	0	0	
K-G	0	0	74	0	0	0	0	0	0	0	0	0	0	
K-H	0	0	71	0	1	0	0	0	0	0	0	6	0	
K-I	26	0	80	81	155	0	1	3	0	3	8	57	0	
K-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
K-K	0	0	0	0	0	1	0	0	0	0	0	0	0	
K-L	0	0	0	0	0	0	0	0	0	182	0	0	0	
K-M	0	0	0	0	0	0	0	0	0	0	0	1	0	
K-N	0	0	0	0	0	13	0	0	0	0	0	0	0	
K-O	13	0	53	6	7	23	23	26	0	1	1	6	0	
K-P	0	0	0	0	0	0	0	0	0	0	0	0	0	
K-Q	0	0	0	0	0	0	0	0	0	0	0	0	0	
K-R	0	0	0	0	0	24	0	0	0	4	0	0	0	
K-S	3	0	0	0	47	0	0	2	0	3	0	1499	0	
K-T	0	0	0	0	0	2	0	0	0	0	0	1	0	
K-U	33	0	3	2	9	0	0	0	0	3	0	8	0	
K-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
K-W	0	0	0	0	0	2	0	0	0	0	0	0	0	
K-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
K-Y	0	0	20	3	1	0	0	33	0	0	0	271	0	
K-Z	0	0	0	0	0	0	0	0	0	0	0	0	0	
K-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
K-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
L-A-	4	701	1641	348	12	21	223	27	864	0	84	34	357	3032	89
L-B-	28	0	0	0	59	0	0	0	6	0	0	3	0	0	64
L-C-	15	0	0	0	14	0	0	38	17	0	1	3	0	0	131
L-D-	13	15	5	1	651	1	3	51	535	0	0	83	2	10	56
L-E-	3037	82	1408	2966	340	511	1010	13	64	1	3	86	1156	1340	56
L-F-	57	11	1	0	26	2	1	1	59	0	0	3	0	0	10
L-G-	32	0	0	0	51	0	0	25	70	0	0	2	0	0	5
L-H-	20	0	0	0	12	0	0	0	6	0	0	0	0	0	38
L-I-	825	447	2201	372	1681	1104	1518	15	0	1	1734	48	634	3748	458
L-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L-K-	22	0	0	0	279	0	0	5	186	0	0	29	0	23	2
L-L-	767	20	8	1	2951	22	2	3	1814	0	1	0	44	40	1484
L-M-	104	0	0	1	179	0	0	0	68	0	0	11	0	3	481
L-N-	0	0	0	0	120	0	3	0	1	0	0	0	0	0	1
L-O-	307	110	985	120	23	13	858	4	68	0	3	22	94	2066	1702
L-P-	33	0	0	0	88	34	0	134	51	0	0	37	1	0	17
L-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L-R-	14	0	0	0	274	0	0	0	4	0	0	0	0	0	90
L-S-	12	12	0	1	312	1	0	17	107	0	21	1	1	0	1186
L-T-	186	2	0	5	476	0	0	540	444	0	0	2	6	6	146
L-U-	135	231	234	589	850	16	85	0	61	2	10	35	629	296	28
L-V-	139	0	0	4	381	0	0	0	93	0	0	0	0	0	3
L-W-	501	0	0	0	2	0	0	1	4	0	0	0	0	0	18
L-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L-Y-	36	4	23	3	31	7	1	11	178	0	2	8	51	90	8
L-Z-	3	3	0	0	2	0	0	0	5	0	0	0	1	0	0
L-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
L-A	141	6	2528	1892	3087	375	188	584	72	1125	77	359	0	
L-B	0	0	10	9	0	21	0	0	0	1	0	26	0	
L-C	0	0	1	0	0	82	0	0	0	6	0	0	0	
L-D	0	0	368	197	2	5	3	13	0	2	0	8915	0	
L-E	189	0	980	3550	1625	34	608	115	299	364	5	10963	0	
L-F	0	0	86	2	6	25	0	19	0	0	0	1682	0	
L-G	0	0	20	0	0	0	0	0	0	0	0	0	0	
L-H	0	0	0	0	0	0	0	0	0	0	0	1	0	
L-I	308	127	8	1916	3646	31	862	0	34	0	702	90	0	
L-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
L-K	0	0	0	71	0	0	0	2	0	20	0	400	0	
L-L	37	0	8	722	2	257	2	22	0	3552	0	10926	0	
L-M	0	0	1	45	0	7	0	0	0	3	0	199	0	
L-N	0	0	0	0	0	21	0	0	0	0	0	49	0	
L-O	861	24	873	1314	631	539	509	2253	1	371	0	158	0	
L-P	0	0	9	36	33	0	0	0	0	2	0	327	0	
L-Q	0	0	0	0	0	4	0	0	0	0	0	0	0	
L-R	0	0	0	0	0	1	0	0	0	39	0	0	0	
L-S	1	0	0	3	50	0	2	1	0	2	0	2939	0	
L-T	0	0	61	248	0	215	0	0	0	267	17	1237	0	
L-U	4	0	157	642	493	1	2	0	81	0	1	19	0	
L-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
L-W	0	0	0	0	0	0	0	0	0	1	0	0	0	
L-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
L-Y	20	0	42	182	51	2	0	36	0	0	49	16348	0	
L-Z	0	0	1	0	0	1	0	0	0	0	0	1	0	
L-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
L-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
M-A-	0	49	414	1282	27	0	781	51	0	337	1302	1449	85	5696	7
M-B-	157	3	0	0	1905	0	0	0	346	0	0	423	0	3	220
M-C-	1	0	2	0	1	0	0	8	6	0	0	0	0	0	7
M-D-	2	0	0	0	2	0	0	0	0	0	0	0	0	0	1
M-E-	1376	63	135	1980	403	5	28	74	9	7	3	387	988	7993	102
M-F-	0	0	0	0	5	0	1	0	1	0	0	0	0	0	138
M-G-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M-H-	2	0	0	0	1	0	0	0	0	0	0	0	0	0	13
M-I-	134	0	955	303	109	6	764	0	0	6	95	1998	22	3688	2
M-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M-K-	0	0	0	0	0	0	0	0	5	0	0	0	0	0	1
M-L-	2	0	0	0	29	0	0	0	23	0	0	0	0	0	3
M-M-	361	0	0	0	1026	0	0	0	582	0	1	1	4	0	456
M-N-	38	0	0	0	59	0	0	0	33	0	0	9	0	0	7
M-O-	7	164	231	612	2	4	46	3	58	0	74	213	361	2254	262
M-P-	1219	4	0	0	761	3	5	254	229	0	9	2306	4	6	1162
M-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M-R-	18	0	0	0	0	0	0	0	8	0	0	0	0	0	5
M-S-	1	2	6	1	889	0	0	5	6	0	1	0	2	0	13
M-T-	0	0	0	0	3	0	0	8	0	0	1	0	0	0	1
M-U-	0	1	952	67	38	16	18	1	1	0	1	478	189	744	1
M-V-	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0
M-W-	1	0	0	0	22	0	0	0	0	0	0	0	0	0	1
M-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M-Y-	0	0	11	0	6	0	0	0	0	0	0	2	0	5	12
M-Z-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
M-A	47	1	2232	665	2456	43	6	6	147	1621	69	369	0	:
M-B	3	0	97	67	0	77	0	0	0	3	0	118	0	:
M-C	0	0	0	0	0	0	0	0	0	0	0	1	0	:
M-D	0	0	1	0	0	2	0	0	0	0	0	7	0	:
M-E	8	0	2660	1601	1762	1	6	204	57	49	3	9280	0	:
M-F	0	0	0	0	0	7	0	0	0	0	0	0	0	:
M-G	0	0	0	0	0	0	0	0	0	0	0	17	0	:
M-H	0	0	0	0	0	1	0	0	0	0	0	0	0	:
M-I	1	3	181	1628	1225	27	0	0	102	2	73	66	0	:
M-J	0	0	0	0	0	0	0	0	0	0	0	0	0	:
M-K	0	0	0	0	0	0	0	0	0	0	0	0	0	:
M-L	0	0	0	0	0	0	0	0	0	106	0	11	0	:
M-M	0	0	0	17	0	707	0	0	0	45	0	60	0	:
M-N	0	0	0	45	0	0	0	0	0	1	0	162	0	:
M-O	20	0	3577	1901	945	877	956	1	1	5	13	33	0	:
M-P	1	0	531	161	583	210	0	0	0	8	0	238	0	:
M-Q	0	0	0	0	0	0	0	0	0	0	0	0	0	:
M-R	0	0	0	534	0	0	0	0	0	0	0	839	0	:
M-S	7	0	0	1	119	2	0	0	0	8	0	2378	0	:
M-T	0	0	0	1	0	0	0	0	0	0	0	9	0	:
M-U	1	0	219	1710	141	0	0	0	0	0	14	1	0	:
M-V	0	0	0	0	0	0	0	0	0	0	0	0	0	:
M-W	0	0	0	0	0	0	0	0	0	0	0	0	0	:
M-X	0	0	0	0	0	0	0	0	0	0	0	0	0	:
M-Y	0	0	38	224	78	0	0	0	0	0	0	1807	0	:
M-Z	0	0	0	0	0	0	0	0	0	0	0	0	0	:
M-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	:
M-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
N-A-	0	339	176	157	26	17	385	16	150	0	133	3536	674	618	3
N-B-	18	0	0	0	64	0	0	0	2	0	0	5	0	0	18
N-C-	71	0	0	0	8426	0	0	967	1230	0	11	756	0	0	644
N-D-	1004	37	11	2	4012	39	6	12	2631	0	10	445	77	25	627
N-E-	692	20	663	3677	1016	146	358	11	327	0	13	443	222	357	141
N-F-	114	0	0	0	298	0	0	0	291	0	0	304	0	0	619
N-G-	206	8	0	31	2399	52	1	57	575	0	1	1112	4	19	105
N-H-	21	0	0	0	96	0	0	1	45	0	0	0	0	0	79
N-I-	428	22	1132	4	426	527	646	12	1	1	39	56	326	3118	636
N-J-	15	0	0	0	27	0	0	0	10	0	0	0	0	0	164
N-K-	24	0	1	0	268	41	1	3	340	0	0	114	7	54	11
N-L-	52	0	0	0	193	0	0	0	160	0	0	0	0	0	46
N-M-	38	0	0	0	744	0	0	0	18	0	0	0	0	0	11
N-N-	259	0	0	0	1242	1	1	2	771	0	0	0	0	5	597
N-O-	1	143	161	159	21	17	32	0	113	0	1	175	738	415	161
N-P-	30	0	0	0	3	0	0	2	27	0	0	42	0	0	42
N-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N-R-	19	0	0	0	87	0	0	0	95	0	0	0	0	0	43
N-S-	205	2	274	17	1470	165	2	247	2336	0	20	95	51	0	367
N-T-	1944	1	3	0	5066	6	16	714	4215	0	0	1126	82	6	2150
N-U-	247	10	171	32	420	132	15	1	149	0	1	25	812	25	96
N-V-	146	0	0	0	743	0	0	0	385	0	0	0	0	0	311
N-W-	67	0	0	0	28	0	0	40	63	0	0	0	0	0	12
N-X-	0	0	0	0	1	0	0	0	81	0	0	0	0	0	0
N-Y-	5	48	0	1	2	0	1	20	31	0	0	12	45	0	162
N-Z-	36	0	0	0	23	0	0	0	6	0	0	0	0	0	3
N-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY
N-A	113	0	672	141	2981	55	118	33	3	10	30	506	0
N-B	0	0	16	0	0	37	0	0	0	0	0	0	0
N-C	0	0	643	6	505	65	0	0	0	447	1	35	0
N-D	19	1	358	1617	20	698	1	32	0	120	0	32000	0
N-E	23	16	2507	2975	493	151	933	2477	486	568	9	7916	0
N-F	0	0	117	0	0	188	0	0	0	0	0	1	0
N-G	2	0	457	1583	635	409	1	7	0	15	0	29538	0
N-H	0	0	0	0	0	18	0	1	0	2	0	3	0
N-I	38	238	10	1402	2039	54	418	0	34	0	603	95	0
N-J	0	0	0	0	0	158	0	0	0	0	0	0	0
N-K	1	0	15	253	2	0	0	0	0	7	0	1081	0
N-L	0	0	0	0	0	3	0	0	0	2221	0	0	0
N-M	0	0	0	0	0	3	0	0	0	1	0	0	0
N-N	0	0	0	54	1	121	0	0	0	193	0	131	0
N-O	47	0	1356	172	6689	746	331	2769	16	27	9	2323	0
N-P	0	0	36	0	0	24	0	0	0	0	0	0	0
N-Q	0	0	0	0	0	186	0	0	0	0	0	1	0
N-R	0	0	0	0	0	8	0	0	0	93	0	0	0
N-S	301	0	1	9	2418	432	12	281	0	60	0	7843	0
N-T	7	0	2145	3055	0	487	0	5	0	382	1	13001	0
N-U	5	0	78	132	388	7	1	0	1	0	1	14	0
N-V	0	0	0	0	0	10	0	0	0	7	0	0	0
N-W	0	0	1	0	0	0	0	0	0	0	0	16	0
N-X	0	0	0	0	0	0	0	0	0	0	0	16	0
N-Y	1	0	0	8	281	0	3	86	0	0	0	3433	0
N-Z	0	0	0	0	0	0	0	1	0	27	0	6	0
N-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0
N-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
O-A-	0	0	305	765	0	6	24	3	0	0	59	181	90	149	0
O-B-	406	186	1	1	340	0	0	0	233	320	0	688	0	5	96
O-C-	809	0	555	0	700	0	0	111	1109	0	1163	58	0	7	89
O-D-	327	13	3	182	955	2	105	6	402	0	4	28	12	21	130
O-E-	3	12	3	44	1	10	3	4	7	0	0	38	138	16	1
O-F-	29	0	0	0	261	1743	0	0	141	0	0	5	1	1	55
O-G-	87	1	0	3	447	0	62	4	471	0	0	31	16	248	21
O-H-	11	0	0	1	37	0	0	0	69	0	0	26	37	507	25
O-I-	6	3	438	222	2	1	10	0	0	3	1	362	0	1782	0
O-J-	0	0	0	0	210	0	0	0	0	0	0	0	0	0	10
O-K-	32	0	3	0	992	3	0	0	278	0	13	50	1	2	50
O-L-	461	7	15	2423	1251	68	19	14	1777	0	102	2032	83	56	1192
O-M-	811	411	0	2	6221	133	0	3	1384	0	1	5	2125	9	331
O-N-	2863	1	1741	2071	6505	741	3007	6	842	67	48	1815	97	413	815
O-O-	0	8	5	2155	7	145	9	0	2	0	2173	1247	720	625	12
O-P-	98	0	3	0	2645	7	2	397	459	0	11	908	393	1	427
O-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O-R-	1188	153	796	2372	5346	29	937	81	1787	0	1865	883	2466	932	260
O-S-	152	13	115	10	3275	0	1	36	1028	0	13	32	49	0	201
O-T-	454	42	28	1	1598	2	9	4864	943	0	0	60	51	6	325
O-U-	6	496	244	222	19	9	3740	0	123	0	10	5490	9	5010	0
O-V-	195	0	0	1	5453	0	0	0	1125	0	1	0	0	0	37
O-W-	597	34	3	170	2200	5	3	38	624	0	0	388	26	3224	5
O-X-	8	0	11	0	37	19	0	3	181	0	0	0	1	1	3
O-Y-	167	11	34	19	315	3	0	5	59	0	0	6	87	11	11
O-Z-	10	0	0	0	116	0	0	0	6	0	0	0	0	0	18
O-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
O-A	29	1	481	142	476	0	2	0	7	0	2	25	0	
O-B	0	0	0	414	189	40	208	1	0	9	0	327	0	
O-C	0	0	246	0	306	174	0	0	0	10	2	42	0	
O-D	1	0	47	424	1	840	0	21	0	627	0	2519	0	
O-E	0	0	33	778	270	0	27	1	14	5	0	143	0	
O-F	0	0	0	15	521	11	0	0	0	0	0	32000	0	
O-G	0	0	1000	87	10	60	0	1	0	253	0	141	0	
O-H	0	0	4	0	0	0	0	0	0	2	0	124	0	
O-I	0	0	44	251	70	0	0	0	1	0	1	5	0	
O-J	0	0	0	0	0	1	0	0	0	0	0	0	0	
O-K	0	0	0	223	0	19	0	2	0	28	0	1241	0	
O-L	28	2	3	375	116	834	447	2	0	214	3	1244	0	
O-M	2664	0	17	191	2	10	0	22	0	147	0	5669	0	
O-N	26	29	46	7620	3177	58	549	23	9	319	26	26329	0	
O-O	279	0	845	239	519	0	17	0	0	0	8	864	0	
O-P	837	0	160	229	200	312	0	1	0	65	0	774	0	
O-Q	0	0	0	0	0	40	0	0	0	0	0	0	0	
O-R	389	16	706	1624	4342	44	12	152	0	1198	3	17766	0	
O-S	423	14	2	1607	2940	49	2	1	0	21	0	348	0	
O-T	2	0	30	252	623	15	0	7	0	30	0	6251	0	
O-U	766	5	5068	4168	6449	0	25	0	12	0	1	3322	0	
O-V	0	0	0	12	0	0	0	0	0	4	0	94	0	
O-W	5	0	5	610	158	15	0	1	0	7	1	4862	0	
O-X	0	0	0	0	0	0	1	1	1	66	0	159	0	
O-Y	0	0	2	210	4	0	0	0	0	0	0	499	0	
O-Z	0	0	0	0	0	0	0	0	0	8	15	6	0	
O-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
O-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P-A-	0	128	695	48	4	0	211	10	823	4	14	487	45	1963	3
P-B-	1	0	0	0	6	0	0	0	0	0	0	0	0	0	6
P-C-	0	0	0	0	1	0	0	3	1	0	0	0	0	0	4
P-D-	5	0	0	0	0	0	0	0	0	0	0	0	0	0	2
P-E-	1504	5	2067	1292	305	26	23	1	10	0	2	261	8	2257	890
P-F-	5	0	1	0	1	0	0	0	2	0	0	0	0	0	3
P-G-	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2
P-H-	340	0	0	0	461	0	0	0	704	0	0	19	0	1	603
P-I-	162	0	801	186	470	4	42	0	0	0	76	316	13	1126	96
P-J-	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P-K-	1	0	0	0	8	0	0	0	23	0	0	0	0	0	0
P-L-	3644	0	0	0	3295	0	0	0	1019	0	0	0	0	0	627
P-M-	20	0	0	0	584	0	0	0	0	0	0	0	0	0	6
P-N-	0	0	0	0	15	0	0	0	2	0	0	0	0	0	5
P-O-	5	1	105	40	401	4	16	13	909	0	189	1396	22	1526	307
P-P-	248	0	0	0	1796	0	0	0	380	0	0	731	5	0	1052
P-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P-R-	598	0	0	0	4823	0	0	0	2175	0	0	0	0	0	7458
P-S-	24	0	1	0	119	0	2	18	37	0	3	0	1	2	64
P-T-	212	0	8	0	410	0	0	74	891	0	0	52	0	4	107
P-U-	1	864	4	19	27	11	4	0	2	0	1	686	32	110	0
P-V-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P-W-	34	0	0	0	0	0	0	0	3	0	0	0	0	0	1
P-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P-Y-	1	1	1	0	1	0	0	1	11	0	2	9	0	1	2
P-Z-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
P-A	392	6	4041	1021	1264	134	36	26	9	413	0	86	0	
P-B	0	0	1	0	0	3	0	0	0	0	0	0	0	
P-C	0	0	0	0	0	0	0	0	0	0	0	0	0	
P-D	0	0	1	0	0	0	0	0	0	0	0	0	0	
P-E	30	0	6244	335	548	16	4	20	4	4	7	1049	0	
P-F	0	0	2	0	0	39	0	0	0	0	0	8	0	
P-G	0	0	12	0	0	0	0	0	0	0	0	0	0	
P-H	0	0	100	40	2	10	0	0	0	489	1	205	0	
P-I	74	4	573	126	745	18	4	1	1	0	5	51	0	
P-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
P-K	0	0	0	0	0	0	0	0	0	0	0	0	0	
P-L	0	0	0	0	0	218	0	0	0	569	0	2	0	
P-M	0	0	0	0	0	0	0	0	0	0	0	5	0	
P-N	0	0	0	0	0	0	0	0	0	0	0	0	0	
P-O	385	0	2537	2761	475	238	25	580	4	0	5	21	0	
P-P	0	0	741	5	0	2	0	0	0	154	0	16	0	
P-Q	0	0	0	0	0	0	0	0	0	0	0	0	0	
P-R	0	0	0	0	0	41	0	0	0	6	0	0	0	
P-S	0	0	0	0	50	14	0	2	0	166	0	1440	0	
P-T	0	0	5	102	0	144	0	0	0	69	0	940	0	
P-U	67	0	716	191	804	0	0	0	0	1	44	0	0	
P-V	0	0	0	0	1	0	0	0	0	0	0	0	0	
P-W	0	0	3	0	0	0	0	0	0	0	0	0	0	
P-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
P-Y	0	0	18	2	15	0	0	1	0	0	0	273	0	
P-Z	0	0	0	0	0	0	0	0	0	0	0	0	0	
P-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
P-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q-A-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-B-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-C-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-D-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-E-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-F-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-G-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-H-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-I-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-K-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-L-	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-M-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-N-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-O-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-P-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-R-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-S-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-T-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-U-	1261	0	0	0	1665	0	0	0	1846	0	0	0	0	0	130
Q-V-	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Q-W-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-Y-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-Z-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
Q-A	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-B	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-C	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-D	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-E	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-F	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-G	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-H	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-I	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-K	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-L	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-M	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-N	0	0	0	1	0	0	0	0	0	0	0	3	0	
Q-O	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-P	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-Q	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-R	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-S	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-T	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-U	0	0	0	0	0	0	0	0	0	3	0	0	0	
Q-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-W	0	0	0	0	0	0	0	0	0	0	0	2	0	
Q-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-Y	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-Z	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
Q-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
R-A-	0	621	2064	1349	30	255	853	71	1543	4	40	2927	1185	3104	38
R-B-	197	0	0	0	175	0	0	0	154	0	0	39	0	0	162
R-C-	51	0	0	0	1310	0	0	1459	253	0	2	113	1	0	146
R-D-	213	10	0	0	1258	0	0	1	643	0	0	180	6	11	56
R-E-	7532	203	3219	5130	2771	1236	1108	153	423	111	7	2072	1879	3001	242
R-F-	316	0	0	0	206	0	0	0	29	0	0	16	1	0	286
R-G-	567	0	0	1	1685	0	0	34	371	0	0	20	0	0	125
R-H-	339	0	0	0	60	0	0	0	18	0	0	0	0	0	208
R-I-	1702	882	2756	708	3024	396	1467	0	1	10	137	591	816	3963	1340
R-J-	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1
R-K-	101	3	0	1	841	0	0	13	254	0	0	23	17	47	11
R-L-	114	3	0	813	358	0	0	1	454	0	0	0	3	0	109
R-M-	1412	3	6	0	759	5	0	13	885	0	0	73	0	4	317
R-N-	627	12	0	0	1397	10	1	5	958	0	1	6	512	2	281
R-O-	1006	1242	950	973	107	430	900	58	147	210	245	937	5177	1558	1074
R-P-	37	0	0	0	154	0	0	56	30	0	0	123	5	1	488
R-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-R-	444	0	0	0	1134	0	0	22	1243	0	0	0	1	0	566
R-S-	207	9	12	40	1647	1	0	473	595	0	22	1	2	2	861
R-T-	1454	7	8	0	1200	12	22	1553	1878	0	1	217	403	73	238
R-U-	47	92	767	125	312	44	237	3	200	14	2	302	320	582	3
R-V-	349	0	0	0	926	0	0	0	857	0	0	0	0	0	55
R-W-	224	0	0	0	23	0	0	32	113	0	0	0	0	0	23
R-X-	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0
R-Y-	29	72	8	15	13	2	4	0	349	0	6	56	14	6	126
R-Z-	7	0	0	0	2	1	0	0	2	0	6	0	0	0	7
R-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
R-A	765	15	523	555	4968	67	383	407	5	421	98	469	0	
R-B	0	0	8	41	0	29	0	0	0	0	0	84	0	
R-C	0	0	76	8	7	263	0	0	0	25	0	45	0	
R-D	0	0	13	989	9	9	0	19	0	70	0	3346	0	
R-E	1859	728	248	8709	1706	68	1170	404	32	85	8	21453	0	
R-F	0	0	12	3	0	212	0	0	0	0	0	18	0	
R-G	0	0	49	12	0	178	0	0	0	135	0	143	0	
R-H	0	0	0	0	0	2	0	0	0	53	0	2	0	
R-I	547	6	0	1942	2513	86	1085	2	3	1	289	95	0	
R-J	0	0	0	0	0	3	0	0	0	0	0	0	0	
R-K	3	0	4	355	3	0	0	8	0	15	0	1757	0	
R-L	1	0	0	150	0	10	0	2	0	1243	0	306	0	
R-M	3	0	0	598	28	186	0	0	0	142	0	1220	0	
R-N	25	0	1	287	9	8	0	4	0	6	0	1830	0	
R-O	1618	14	220	1109	882	4065	1406	1301	132	225	40	349	0	
R-P	0	0	366	127	27	10	0	0	0	1	0	100	0	
R-Q	0	0	0	0	0	33	0	0	0	0	0	0	0	
R-R	0	0	0	2	0	95	1	0	0	530	0	37	0	
R-S	39	0	0	6	1969	172	1	2	0	34	0	9064	0	
R-T	0	0	94	877	0	506	0	14	0	624	10	3249	0	
R-U	210	0	58	874	265	0	2	2	3	1	8	21	0	
R-V	0	0	0	0	0	0	0	0	0	1	0	9	0	
R-W	0	0	11	0	0	0	0	0	0	2	0	0	0	
R-X	0	0	0	0	0	0	0	0	0	0	0	8	0	
R-Y	12	0	0	78	190	8	0	53	0	0	0	7361	0	
R-Z	0	0	0	0	1	0	0	0	0	1	0	3	0	
R-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
R-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
S-A-	15	94	198	165	0	148	245	4	2086	0	67	815	958	760	0
S-B-	168	0	0	0	21	0	0	0	1	0	0	0	0	0	28
S-C-	637	0	0	0	438	0	0	1209	639	0	0	114	0	0	867
S-D-	147	0	0	0	14	0	0	0	44	0	0	0	0	0	45
S-E-	875	75	1477	3211	2335	78	110	41	58	0	9	2385	330	2737	8
S-F-	105	0	0	0	104	0	0	0	58	0	0	0	0	0	74
S-G-	3	0	0	0	2	0	0	0	14	0	0	0	0	0	2
S-H-	1368	8	71	0	4474	13	0	2	1840	0	2	40	157	24	3051
S-I-	460	1059	1030	2490	201	113	1253	1	0	0	1	462	788	3159	2989
S-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
S-K-	41	0	0	0	575	0	0	0	485	15	0	5	2	1	28
S-L-	730	0	0	0	316	0	0	0	408	0	0	0	0	0	359
S-M-	893	0	0	0	220	0	0	0	349	0	0	0	0	0	200
S-N-	179	0	0	0	139	0	0	0	23	0	0	0	0	0	133
S-O-	59	56	1077	48	11	150	4	7	90	9	9	983	2944	2735	233
S-P-	663	1	0	0	2704	0	0	211	997	0	0	253	0	2	1280
S-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S-R-	23	0	0	0	28	0	0	1	1	0	0	0	0	0	40
S-S-	755	2	1	0	2173	145	0	12	3647	0	0	77	108	43	592
S-T-	7074	6	14	1	4671	19	3	65	4821	0	0	243	185	14	2724
S-U-	586	1066	1715	206	401	276	307	4	263	0	8	685	761	397	37
S-V-	0	0	0	0	15	0	0	0	26	0	0	0	0	0	0
S-W-	125	0	0	0	579	0	0	0	311	0	0	0	0	0	75
S-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S-Y-	0	3	154	5	1	0	1	0	0	0	0	72	376	98	0
S-Z-	0	0	0	0	2	0	0	0	0	0	0	0	0	0	15
S-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
S-A	146	0	447	123	642	100	239	385	39	867	0	92	0	
S-B	0	0	3	0	0	68	0	0	0	21	0	0	0	
S-C	0	0	869	7	0	395	0	0	0	0	0	6	0	
S-D	0	0	0	0	0	12	0	0	0	0	0	0	0	
S-E	351	198	2253	2210	950	73	680	140	195	94	1	10294	0	
S-F	0	0	2	0	0	149	0	0	0	29	0	0	0	
S-G	0	0	8	0	0	31	0	0	0	0	0	0	0	
S-H	1	0	125	0	2	144	7	8	0	35	0	1630	0	
S-I	77	4	312	1385	2279	18	683	0	321	0	240	16	0	
S-J	0	0	0	0	0	2	0	0	0	0	0	0	0	
S-K	0	0	0	70	0	10	0	0	0	197	0	390	0	
S-L	0	0	0	0	0	89	0	0	0	633	0	1	0	
S-M	0	0	0	64	0	23	0	0	0	4	0	731	0	
S-N	0	0	0	0	0	16	0	0	0	2	0	3	0	
S-O	230	0	734	4	31	1259	208	13	14	13	0	3124	0	
S-P	0	0	456	8	0	108	0	0	0	14	0	41	0	
S-Q	0	0	0	0	0	379	0	0	0	0	0	4	0	
S-R	0	0	0	3	0	22	0	0	0	0	0	4	0	
S-S	6	0	40	3	5	939	0	16	0	51	0	5399	0	
S-T	26	0	5097	1246	0	1340	1	39	0	240	0	12481	0	
S-U	1128	0	2149	367	8	0	6	0	0	0	5	12	0	
S-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
S-W	0	0	4	0	0	49	0	0	0	0	0	1	0	
S-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
S-Y	0	0	16	596	7	0	0	0	0	0	0	399	0	
S-Z	0	0	0	0	0	0	0	0	0	0	0	1	0	
S-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
S-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
T-A-	0	1146	568	28	0	139	661	16	2119	0	1357	2280	153	3126	14
T-B-	57	0	0	0	6	0	0	0	4	0	0	1	0	0	22
T-C-	50	0	0	0	9	0	0	1324	0	0	0	12	0	0	50
T-D-	3	0	0	0	0	0	0	0	4	0	0	0	0	0	44
T-E-	870	12	728	8468	900	52	222	12	115	0	7	1699	1793	3809	77
T-F-	15	0	0	0	1	0	0	0	41	0	0	3	0	0	99
T-G-	23	0	0	0	2	0	0	0	0	0	0	0	0	1	24
T-H-	12736	15	12	49	32000	53	1	18	8963	0	1	115	69	20	4205
T-I-	1103	124	4301	55	1836	693	302	2	0	1	2	1745	3016	5618	17349
T-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
T-K-	2	0	0	0	0	0	0	0	10	0	0	0	0	0	2
T-L-	153	0	0	0	1991	0	0	0	136	0	0	0	0	0	48
T-M-	94	0	0	0	818	0	0	0	25	0	0	0	0	0	141
T-N-	10	0	0	0	240	0	0	0	29	0	0	0	0	0	12
T-O-	32	82	305	307	60	17	399	0	49	1	34	630	840	1562	1714
T-P-	2	0	0	0	9	0	0	2	4	0	0	0	0	0	29
T-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T-R-	4751	0	0	0	1984	0	0	0	2800	0	0	0	0	0	1997
T-S-	4	31	11	1	328	5	0	15	232	0	14	1	34	0	70
T-T-	431	0	0	0	3441	0	1	5	950	0	0	1389	1	0	297
T-U-	1018	137	70	1122	149	49	14	2	48	0	0	64	186	418	45
T-V-	3	0	0	0	0	0	0	0	2	0	0	0	0	0	0
T-W-	46	0	0	0	924	0	0	3	211	0	0	0	0	0	1480
T-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T-Y-	14	2	3	0	1	0	1	0	7	0	16	137	6	0	0
T-Z-	2	4	0	0	44	0	6	3	7	0	5	0	1	0	1
T-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY
T-A	187	0	1661	298	3495	142	26	22	345	235	0	506	0
T-B	0	0	12	0	0	10	0	0	0	0	0	0	0
T-C	0	0	7	0	0	3	0	0	0	8	0	58	0
T-D	0	0	2	0	0	0	0	0	0	0	0	5	0
T-E	371	0	12399	2702	69	47	173	49	307	10	0	7027	0
T-F	0	0	1	0	0	144	0	0	0	0	0	0	0
T-G	0	0	9	0	0	9	0	0	0	0	0	0	0
T-H	20	18	2342	356	9	537	0	88	0	345	0	11601	0
T-I	143	47	509	973	1284	9	2885	0	0	0	183	66	0
T-J	0	0	0	0	0	0	0	0	0	0	0	0	0
T-K	0	0	0	0	0	1	0	0	0	0	0	0	0
T-L	0	0	0	0	0	0	0	0	0	1943	0	1	0
T-M	0	0	0	0	0	0	0	0	0	0	0	0	0
T-N	0	0	0	0	0	13	0	0	0	0	0	0	0
T-O	668	0	3692	87	307	434	24	899	8	26	8	28290	0
T-P	0	0	1	0	0	41	0	0	0	0	0	0	0
T-Q	0	0	0	0	0	0	0	0	0	0	0	0	0
T-R	0	0	0	0	0	1636	0	0	0	1089	1	0	0
T-S	8	0	0	0	62	61	2	6	0	3	0	10739	0
T-T	0	0	174	88	0	8	2	1	0	194	1	196	0
T-U	53	0	4318	180	476	0	0	0	10	0	0	10	0
T-V	0	0	0	0	0	0	0	0	0	0	0	0	0
T-W	0	0	0	0	0	3	0	0	0	0	0	0	0
T-X	0	0	0	0	0	0	0	0	0	0	0	0	0
T-Y	475	0	51	6	0	5	0	1	0	0	0	6377	0
T-Z	0	0	1	2	0	0	0	0	0	0	0	70	0
T-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0
T-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
U-A-	0	56	42	91	0	1	154	0	40	0	33	2269	14	155	0
U-B-	93	153	15	25	123	1	9	6	66	298	2	1189	122	1	33
U-C-	340	0	393	0	572	0	0	2507	200	0	515	169	0	0	19
U-D-	47	3	1	310	1137	0	341	1	709	0	2	32	2	1	30
U-E-	15	8	2	324	93	5	1	0	0	0	0	169	7	674	14
U-F-	125	0	0	0	0	526	0	0	0	0	0	5	0	1	0
U-G-	99	0	0	0	178	0	492	4152	9	0	0	71	12	3	22
U-H-	3	0	0	0	0	0	0	4	1	0	0	1	2	3	0
U-I-	4	6	225	251	156	1	1	0	0	0	0	647	1	277	2
U-J-	2	0	0	0	0	0	0	0	17	0	0	0	0	0	5
U-K-	8	0	0	0	42	0	0	6	2	0	1	1	0	0	0
U-L-	1671	17	23	5354	402	91	48	0	204	0	71	1108	47	65	99
U-M-	460	955	2	3	889	6	0	1	172	0	0	6	301	151	126
U-N-	366	59	1118	4728	394	163	870	69	2684	8	236	255	48	319	34
U-O-	0	0	0	1	0	0	0	1	6	0	1	3	1	2	0
U-P-	71	6	2	4	302	3	6	35	164	0	6	200	1	0	514
U-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
U-R-	1113	271	874	329	4397	241	336	1	1381	0	84	89	78	1618	255
U-S-	278	151	132	1	3980	4	0	509	1679	0	43	565	8	65	19
U-T-	125	26	121	50	1136	32	15	1292	1449	0	0	108	21	8	391
U-U-	0	0	0	0	0	0	0	0	0	0	0	0	30	1	0
U-V-	10	0	0	0	63	0	0	0	6	0	0	0	0	0	7
U-W-	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
U-X-	5	1	0	0	17	0	0	0	10	0	0	6	0	0	1
U-Y-	0	0	0	0	7	0	0	0	31	0	0	10	0	2	0
U-Z-	2	0	0	0	4	0	0	0	0	0	0	0	0	0	3
U-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
U-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
U-A	0	0	663	26	736	0	7	2	0	1	0	14	0	
U-B	6	0	16	444	269	78	6	8	0	0	0	205	0	
U-C	0	0	24	4	1325	8	0	0	0	46	0	7	0	
U-D	0	0	1	131	0	6	0	6	0	311	1	204	0	
U-E	4	0	139	1493	66	2	1	0	0	2	4	1780	0	
U-F	0	0	1	1	2	1	0	0	0	0	0	3	0	
U-G	0	0	1	59	0	99	0	0	0	0	0	145	0	
U-H	0	0	4	1	1	0	0	0	0	0	0	26	0	
U-I	208	0	629	407	840	2	75	0	6	0	6	7	0	
U-J	0	0	0	0	0	1	0	0	0	0	0	0	0	
U-K	0	0	4	0	2	2	0	0	0	0	0	11	0	
U-L	59	0	0	169	1678	53	6	5	0	155	4	1105	0	
U-M	391	0	0	203	2	61	4	0	0	0	0	812	0	
U-N	98	18	93	353	2467	86	5	78	0	13	1	626	0	
U-O	0	0	69	5	68	185	1	0	0	5	0	48	0	
U-P	906	0	88	234	209	8	0	33	0	25	0	2431	0	
U-Q	0	0	0	0	0	8	0	0	0	0	0	0	0	
U-R	491	3	623	1351	915	4	279	1	0	439	5	3030	0	
U-S	179	1	0	561	3673	380	0	0	0	76	0	4158	0	
U-T	49	0	137	447	401	294	0	22	0	159	4	10002	0	
U-U	0	0	0	0	1	0	0	0	0	0	0	1	0	
U-V	0	0	2	0	0	0	0	0	0	0	0	0	0	
U-W	0	0	0	0	0	0	0	0	0	0	0	0	0	
U-X	0	0	0	0	7	31	0	0	0	0	0	86	0	
U-Y	0	0	0	36	1	3	1	0	0	0	0	122	0	
U-Z	0	0	0	0	0	1	0	0	0	1	93	9	0	
U-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
U-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
V-A-	0	63	144	54	6	0	119	3	340	0	6	1190	6	646	1
V-B-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V-C-	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
V-D-	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
V-E-	112	1	37	1782	9	1	38	98	63	3	0	2018	429	3334	14
V-F-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V-G-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V-H-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V-I-	169	18	1007	1437	808	4	84	0	0	0	6	635	1	1615	670
V-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V-K-	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
V-L-	1	0	0	0	1	0	0	0	0	0	0	0	0	0	3
V-M-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V-N-	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
V-O-	0	0	211	0	2	0	6	0	412	0	48	826	5	17	10
V-P-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V-R-	1	0	0	0	11	0	0	0	1	0	0	0	0	0	8
V-S-	0	0	0	0	0	0	0	2	0	0	16	0	0	0	0
V-T-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V-U-	0	5	0	0	10	0	0	1	0	0	0	58	0	0	0
V-V-	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
V-W-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V-Y-	0	0	0	0	0	0	0	0	4	0	0	1	0	0	0
V-Z-	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
V-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
V-A	20	1	694	185	581	25	0	0	0	3	0	96	0	
V-B	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-C	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-D	0	0	0	0	0	0	0	0	0	0	0	4	0	
V-E	3	0	9584	1775	105	0	2	16	9	146	4	10175	0	
V-F	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-G	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-H	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-I	1	0	309	1114	697	1	208	0	0	1	1	2	0	
V-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-K	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-L	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-M	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-N	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-O	1	0	364	7	294	67	0	26	0	35	0	19	0	
V-P	0	0	0	0	0	0	0	0	0	0	0	2	0	
V-Q	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-R	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-S	0	0	0	0	0	0	0	0	0	0	0	19	0	
V-T	0	0	0	0	0	0	0	0	0	0	0	1	0	
V-U	0	0	3	4	0	0	0	0	0	0	0	1	0	
V-V	0	0	0	0	0	0	0	0	0	1	0	0	0	
V-W	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-Y	0	0	0	0	0	0	0	0	0	0	0	192	0	
V-Z	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
V-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
W-A-	0	13	10	45	0	2	246	3	375	0	85	840	18	769	0
W-B-	8	0	0	0	7	0	0	0	5	0	0	0	0	0	32
W-C-	6	0	0	0	1	0	0	0	0	0	0	0	0	0	14
W-D-	2	0	0	0	88	0	0	0	6	0	0	3	0	0	6
W-E-	564	18	2	784	1397	0	8	0	218	0	0	1345	2	674	0
W-F-	2	0	0	0	1	0	0	0	2	0	0	1	0	0	8
W-G-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W-H-	2176	0	0	0	4095	0	0	0	4961	0	0	0	0	0	3041
W-I-	1	0	186	318	9	283	28	0	0	0	0	2845	56	1649	0
W-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W-K-	0	0	0	0	7	0	0	0	5	0	0	0	0	0	0
W-L-	1	0	0	0	250	0	0	0	33	0	0	0	0	0	0
W-M-	38	0	0	0	10	0	0	0	1	0	0	0	0	0	0
W-N-	1	1	2	2	186	5	4	6	41	0	0	3	0	0	0
W-O-	0	9	0	1	9	4	0	4	0	0	23	47	433	294	338
W-P-	2	0	0	0	0	0	0	2	1	0	0	0	0	0	47
W-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W-R-	54	0	0	0	98	0	0	0	736	0	0	0	0	0	326
W-S-	2	2	0	0	5	0	0	40	3	0	8	7	5	0	7
W-T-	0	0	0	0	0	0	0	165	3	0	0	0	0	0	9
W-U-	0	0	0	0	0	0	0	1	0	0	0	17	2	48	0
W-V-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W-W-	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W-Y-	3	0	5	0	70	0	0	0	0	0	1	2	1	5	9
W-Z-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
W-A	3	0	2254	10245	912	9	187	0	23	2351	2	19	0	
W-B	0	0	1	0	0	11	0	0	0	0	0	0	0	
W-C	0	0	0	0	0	1	0	0	0	0	0	0	0	
W-D	0	0	3	12	0	3	0	0	0	8	0	64	0	
W-E	43	0	4611	562	70	0	552	0	6	9	0	2694	0	
W-F	0	0	0	0	0	32	0	0	0	0	0	0	0	
W-G	0	0	2	0	0	1	0	0	0	0	0	0	0	
W-H	0	0	3	1	0	0	0	0	0	406	0	2	0	
W-I	39	0	99	696	8569	0	32	0	0	0	3	1	0	
W-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
W-K	0	0	0	12	0	0	0	17	0	1	0	15	0	
W-L	0	0	0	17	2	0	0	0	0	153	0	61	0	
W-M	0	0	0	0	0	0	0	0	0	0	0	0	0	
W-N	4	0	0	124	47	3	0	19	0	10	0	2919	0	
W-O	2	0	3445	3	1	2777	16	1	0	0	1	1412	0	
W-P	0	0	0	0	0	1	0	0	0	0	0	0	0	
W-Q	0	0	0	0	0	0	0	0	0	0	0	0	0	
W-R	0	0	0	0	1	0	0	0	0	14	0	2	0	
W-S	110	0	1	1	4	2	0	2	0	1	1	941	0	
W-T	0	0	0	1	0	6	0	0	0	0	0	11	0	
W-U	1	0	5	2	0	0	0	0	0	0	0	1	0	
W-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
W-W	0	0	0	0	0	0	0	1	0	0	0	1	0	
W-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
W-Y	0	0	0	0	0	0	0	0	0	0	0	9	0	
W-Z	0	0	0	0	0	0	0	0	0	1	0	0	0	
W-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
W-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
X-A-	0	8	144	0	0	0	34	0	0	0	0	16	478	67	0
X-B-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
X-C-	14	0	0	0	510	0	0	90	108	0	0	125	0	0	6
X-D-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X-E-	0	0	112	188	0	0	1	0	1	0	0	16	28	18	0
X-F-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19
X-G-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X-H-	38	0	0	0	0	0	0	0	86	0	0	1	0	0	4
X-I-	9	44	85	35	57	0	3	0	1	0	0	18	213	62	61
X-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X-K-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X-L-	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0
X-M-	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
X-N-	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0
X-O-	0	0	0	6	0	0	3	0	0	0	0	0	3	61	0
X-P-	164	0	0	0	1367	0	0	0	13	0	0	446	0	0	117
X-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X-R-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
X-S-	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
X-T-	10	5	0	0	449	0	0	26	74	0	0	0	0	0	13
X-U-	75	11	0	2	0	0	0	0	0	0	0	4	0	0	0
X-V-	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
X-W-	0	0	0	0	11	0	0	0	0	0	0	0	0	0	2
X-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X-Y-	0	0	0	5	0	0	46	1	0	0	0	6	1	0	0
X-Z-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
X-A	5	0	1	79	21	0	1	0	0	0	1	1	0	
X-B	0	0	1	0	0	0	0	0	0	0	0	0	0	
X-C	0	0	3	0	0	40	0	0	0	0	0	0	0	
X-D	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-E	0	0	142	110	4	0	0	0	0	1	0	9	0	
X-F	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-G	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-H	0	0	0	0	0	2	0	0	0	0	0	0	0	
X-I	0	0	0	388	30	0	0	0	0	0	0	16	0	
X-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-K	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-L	0	0	0	0	0	0	0	0	0	2	0	0	0	
X-M	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-N	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-O	5	0	8	1	14	0	0	0	0	0	0	0	0	
X-P	0	0	281	0	0	7	0	0	0	0	0	0	0	
X-Q	0	0	0	0	0	7	0	0	0	0	0	0	0	
X-R	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-S	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-T	0	0	293	8	0	66	0	0	0	21	0	492	0	
X-U	0	0	36	0	0	0	0	0	0	0	0	0	0	
X-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-W	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-X	0	0	0	0	0	0	0	0	2	0	0	2	0	
X-Y	0	0	0	0	3	0	0	0	0	0	0	22	0	
X-Z	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
X-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Y-A-	0	22	16	2	0	0	25	1	0	1	4	153	5	180	0
Y-B-	3	0	0	0	157	0	0	0	2	0	0	0	0	0	127
Y-C-	2	0	0	0	46	0	0	160	7	0	3	59	0	0	25
Y-D-	18	0	0	0	10	0	0	0	8	0	0	0	0	7	2
Y-E-	1675	18	0	513	89	1	3	3	5	0	1	121	0	13	0
Y-F-	2	0	0	0	4	3	0	0	1	0	0	4	0	0	8
Y-G-	1	0	0	0	46	0	0	0	4	0	0	0	3	1	12
Y-H-	0	0	0	0	1	0	0	0	0	0	0	0	0	0	47
Y-I-	0	0	0	4	64	0	0	0	0	0	0	0	0	1132	0
Y-J-	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y-K-	17	0	0	0	10	0	0	0	5	0	0	0	0	2	0
Y-L-	53	1	0	0	155	0	2	0	89	0	7	32	0	0	51
Y-M-	39	165	0	0	282	0	0	0	7	0	0	0	22	36	75
Y-N-	68	9	20	18	101	0	2	3	18	0	2	0	0	12	68
Y-O-	0	0	6	4	0	7	4	0	0	0	21	5	8	454	4
Y-P-	4	0	0	0	382	0	0	18	98	0	0	1	0	4	109
Y-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y-R-	65	0	0	10	20	0	0	0	65	0	0	0	0	2	179
Y-S-	1	2	7	0	162	0	0	0	442	0	0	3	3	0	8
Y-T-	0	0	0	0	33	0	0	629	63	0	0	0	0	0	37
Y-U-	0	3	2	0	0	0	12	2	0	1	6	0	0	2	0
Y-V-	3	0	0	0	2	0	0	0	2	0	0	0	0	0	0
Y-W-	90	0	0	0	4	0	0	86	10	0	0	0	0	0	39
Y-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y-Y-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y-Z-	8	0	0	0	40	0	0	0	8	0	0	0	0	0	0
Y-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
Y-A	1	1	178	5	19	1	0	7	0	0	0	21	0	
Y-B	0	0	7	0	0	51	0	0	0	1	0	0	0	
Y-C	0	0	3	0	0	0	0	0	0	0	0	1	0	
Y-D	88	0	0	0	0	0	0	0	0	0	0	19	0	
Y-E	1	0	367	668	453	0	8	2	0	0	2	158	0	
Y-F	0	0	1	0	0	6	0	0	0	0	0	0	0	
Y-G	0	0	4	0	0	1	0	0	0	1	0	0	0	
Y-H	0	0	1	0	0	0	0	0	0	1	0	0	0	
Y-I	1	0	0	10	0	0	0	0	0	0	0	2	0	
Y-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
Y-K	0	0	0	0	0	0	0	0	0	0	0	5	0	
Y-L	2	0	0	9	1	5	51	0	0	15	0	23	0	
Y-M	209	0	0	5	0	2	0	0	0	2	0	9	0	
Y-N	0	0	0	1	48	0	0	0	1	0	0	45	0	
Y-O	3	0	367	6	11	4899	13	19	2	0	0	25	0	
Y-P	0	0	16	10	33	0	0	0	0	0	0	6	0	
Y-Q	0	0	0	0	0	0	0	0	0	0	0	0	0	
Y-R	0	0	2	1	1	15	0	0	0	0	1	13	0	
Y-S	2	0	0	18	839	5	0	0	0	0	0	1798	0	
Y-T	0	0	1	0	7	0	0	0	0	0	0	11	0	
Y-U	0	0	7	11	0	0	0	0	0	0	0	3	0	
Y-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
Y-W	0	0	6	0	0	0	0	0	0	0	0	0	0	
Y-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
Y-Y	0	0	0	0	0	0	0	0	0	2	0	1	0	
Y-Z	0	0	0	0	0	0	0	0	0	0	0	0	0	
Y-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
Y-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Z-A-	11	64	3	3	1	0	9	1	4	0	3	20	0	24	0
Z-B-	0	0	0	0	3	0	0	0	0	0	0	0	0	0	4
Z-C-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z-D-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z-E-	20	2	6	746	16	1	0	0	4	0	2	12	14	260	0
Z-F-	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Z-G-	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0
Z-H-	1	0	0	0	1	0	0	0	2	0	0	0	0	0	1
Z-I-	2	2	3	2	42	2	6	0	0	0	0	19	4	339	12
Z-J-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z-K-	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0
Z-L-	0	0	0	0	77	0	0	0	26	0	0	0	0	0	1
Z-M-	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Z-N-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z-O-	13	0	0	3	5	0	1	0	1	0	0	16	2	91	19
Z-P-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z-R-	2	0	0	0	0	0	0	0	1	0	0	0	0	0	1
Z-S-	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
Z-T-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z-U-	0	2	0	0	5	0	0	0	0	0	1	0	0	0	0
Z-V-	0	0	0	0	1	0	0	0	0	0	0	0	0	0	7
Z-W-	0	0	0	0	2	0	0	0	0	0	1	0	0	0	2
Z-X-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z-Y-	0	2	1	0	0	0	7	0	0	0	1	0	21	2	0
Z-Z-	39	0	0	0	10	0	0	0	38	0	0	103	1	0	15
Z-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
Z-A	2	0	67	4	509	0	0	0	0	0	0	47	0	
Z-B	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-C	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-D	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-E	0	0	108	99	11	0	0	0	0	1	0	646	0	
Z-F	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-G	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-H	0	0	0	0	0	1	0	0	0	0	0	0	0	
Z-I	3	0	1	13	3	0	0	0	0	0	0	18	0	
Z-J	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-K	0	0	0	0	0	0	0	0	0	4	0	0	0	
Z-L	0	0	0	0	0	0	0	0	0	2	0	0	0	
Z-M	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-N	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-O	6	0	17	3	2	2	5	1	0	0	0	20	0	
Z-P	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-Q	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-R	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-S	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-T	0	0	0	0	0	0	0	0	0	0	0	1	0	
Z-U	0	0	14	1	1	0	0	0	0	0	0	5	0	
Z-V	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-W	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-X	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-Y	0	0	0	0	0	0	0	0	0	0	0	73	0	
Z-Z	0	0	0	0	0	1	0	0	0	0	0	115	0	
Z-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
Z-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
SP-A-	9	3198	3688	2528	113	1794	2452	140	739	2	12	7709	2207	32000	4
SP-B-	4481	0	0	1	18118	0	0	1	1614	0	0	1552	0	0	4166
SP-C-	9060	0	2	0	1989	8	0	5748	1302	0	0	3235	22	0	21963
SP-D-	3047	0	0	0	9136	0	0	1	7324	7	0	0	0	1	5395
SP-E-	2551	13	517	842	6	946	99	9	567	8	31	1510	1184	3958	1
SP-F-	5057	0	0	0	3452	2	0	0	6281	2	0	1538	0	0	14961
SP-G-	1728	0	0	0	2765	0	0	58	1667	0	0	717	12	13	4275
SP-H-	16013	0	0	0	17521	0	0	0	12288	0	0	0	2	0	5820
SP-I-	0	7	70	784	3	2200	104	1	1	1	5	359	2305	32000	87
SP-J-	842	0	0	0	583	0	0	0	83	0	0	0	0	0	1524
SP-K-	267	0	2	0	1073	0	0	81	1189	0	0	21	1	1961	140
SP-L-	4961	23	0	0	5060	0	0	0	7109	0	0	11	0	0	5189
SP-M-	12575	0	3	5	6781	1	17	0	4692	0	0	11	20	0	8449
SP-N-	2300	0	0	1	5943	0	2	0	986	1	2	0	0	0	10944
SP-O-	50	1174	647	112	37	32000	7	158	137	0	42	845	56	12567	6
SP-P-	6633	0	0	0	5053	10	0	1143	1685	0	0	3543	0	3	5802
SP-Q-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SP-R-	3277	0	0	3	15851	1	0	200	2154	0	0	0	0	0	2875
SP-S-	6399	0	2593	0	9163	2	0	8193	4698	0	567	1234	1199	361	8831
SP-T-	3252	0	3	0	3799	0	0	32000	2841	1	0	0	0	0	31054
SP-U-	0	2	0	7	0	0	31	11	0	0	4	148	23	5457	0
SP-V-	1619	0	0	0	1676	0	0	0	2019	0	0	1	0	0	1009
SP-W-	15360	0	0	0	9348	0	0	14225	13037	0	0	0	0	0	7041
SP-X-	1	0	0	0	4	0	0	0	0	0	0	0	0	0	0
SP-Y-	270	0	0	1	2445	2	0	0	77	0	0	0	0	0	5242
SP-Z-	8	0	0	0	97	0	0	3	43	0	0	1	0	0	47
SP-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SP-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE	DUMMY	
SP-A	2346	22	7915	9667	7051	1035	669	799	73	7	7	0	0	
SP-B	0	0	3309	0	0	6927	0	0	0	5360	0	0	0	
SP-C	2	0	2204	0	1	1469	0	0	0	126	10	0	0	
SP-D	0	0	2386	0	0	1370	1	65	0	129	0	0	0	
SP-E	145	555	302	1012	238	247	3446	2	5327	573	2	0	0	
SP-F	0	0	7248	0	14	1791	0	0	0	2	0	0	0	
SP-G	0	0	4433	0	0	1008	0	1	0	78	0	0	0	
SP-H	0	0	7	0	0	1558	0	4	0	270	0	0	0	
SP-I	1	0	334	10745	11162	1	32	0	0	0	4	0	0	
SP-J	0	0	74	0	0	2037	0	0	0	0	0	0	0	
SP-K	0	0	66	0	0	15	0	6	0	12	0	0	0	
SP-L	1	0	0	0	13	601	0	0	0	112	0	0	0	
SP-M	1	0	1373	1	6	3155	0	0	0	1669	0	0	0	
SP-N	0	0	0	0	0	1050	0	2	0	11	0	0	0	
SP-O	1864	0	5915	66	2139	4237	1681	982	118	20	13	0	0	
SP-P	9	0	11849	173	23	2560	1	0	0	38	0	0	0	
SP-Q	0	0	0	0	0	1913	0	0	0	0	0	0	0	
SP-R	5	0	0	0	2	1367	0	0	0	34	0	0	0	
SP-S	3634	312	4	1	11908	6965	3	813	0	1070	16	0	0	
SP-T	0	0	4936	35	0	1144	1	1784	0	473	0	0	0	
SP-U	2688	0	225	2514	203	0	0	0	1	0	0	0	0	
SP-V	2	0	3	16	0	47	0	0	0	3	0	0	0	
SP-W	0	0	1128	0	1	4	0	0	0	23	0	0	0	
SP-X	0	0	0	0	0	0	0	0	1	10	0	0	0	
SP-Y	0	0	4	1	1	28	2	0	0	1	0	0	0	
SP-Z	0	0	0	0	0	9	0	3	0	0	0	0	0	
SP-SP-	0	0	0	0	0	0	0	0	0	0	0	0	0	
SP-DU-	0	0	0	0	0	0	0	0	0	0	0	0	0	

Appendix F. Word List from Dahl Corpus

i	*and*	*the*	*to*	*that*	*you*
of	a	*know*	*was*	*in*	*but*
this	me	*about*	*just*	don't	*my*
i'm	*like*	*or*	have	*so*	it's
think	*be*	*with*	*he*	*well*	do
on	*because*	*really*	*as*	*at*	if
had	all	*she*	*said*	mean	*then*
that's	would	*there*	*very*	*we*	*get*
going	*her*	*up*	*say*	*way*	feel
one	sort	*were*	*want*	didn't	time
your	*they*	*are*	go	see	*can*
him	some	*other*	*why*	*how*	*been*
thought	no	*right*	kind	*here*	*an*
thinking	you're	*from*	them	i've	*maybe*
did	*much*	*could*	can't	*being*	*myself*
even	too	*any*	*little*	*always*	*back*
these	who	*good*	*anything*	last	*by*
felt	mother	*his*	doing	*oh*	*than*
remember	make	*mind*	*into*	*has*	night
saying	*down*	*before*	*went*	*where*	talking
never	i'll	he's	wasn't	same	*only*
dream	first	*whether*	sure	seems	doesn't
lot	*also*	*wanted*	*trying*	around	*feelings*
might	*getting*	having	take	fact	*still*
came	*after*	suppose	*else*	talk	yes
tell	couldn't	*real*	today	*will*	she's
isn't	whole	*work*	*part*	wouldn't	*does*
made	*everything*	off	*used*	*another*	girl
anyway	*though*	told	*probably*	*point*	*look*
away	*understand*	ok	school	*put*	morning
long	afraid	times	*week*	*through*	bad
keep	*started*	reason	*must*	they're	done
almost	*those*	*yet*	*coming*	*nothing*	quite

better	*funny*	*wrong*	*may*	what's	*idea*
find	*able*	*such*	*yourself*	*big*	*happened*
important	*actually*	true	somebody	looking	*give*
guy	*years*	money	let's	next	sometimes
try	our	makes	haven't	nice	*thoughts*
sense	while	either	although	stuff	*own*
hard	*knew*	won't	*call*	life	*exactly*
forth	*let*	*many*	alright	*called*	*their*
friday	certain	*pretty*	*man*	least	except
question	couple	making	*start*	kept	enough
boy	problem	year	*once*	*took*	business
perhaps	bit	ask	*both*	*end*	asked
love	left	*sexual*	situation	*bed*	*old*
between	*place*	talked	stop	certainly	*whatever*
along	relationship	we're	someone	words	ago
says	rather	*analysis*	help	*until*	sex
telling	taking	means	*job*	*gee*	everybody
word	*read*	*reaction*	together	you've	*days*
upset	*hand*	leave	picture	wonder	matter
hour	children	weekend	saturday	saw	*late*
weeks	*particularly*	toward	woman	*child*	*few*
anybody	*care*	*need*	*head*	friends	*mad*
kids	we've	*wanting*	change	*new*	*use*
hadn't	*married*	fantasy	monday	happy	*hell*
family	involved	*show*	who's	*stay*	supposed
clear	*parents*	usually	girls	*wants*	*instead*
guilty	goes	case	*mentioned*	friends	tomorrow
book	finally	sleep	*gets*	*thursday*	*completely*
minutes	*reading*	*answer*	*decided*	*difference*	often
image	obviously	*play*	kid	half	*against*
apparently	*gotten*	shouldn't	*each*	sick	*deal*
gave	tried	anger	strange	*strong*	we'll
particularly	seen	past	*found*	terms	*trouble*

or	*less*	happens	*high*	phone	control
close	hear	realize	somewhere	*reasons*	sister
hours	alone	during	seeing	*women*	*already*
meant	asking	become	conscious	*later*	moment
wife	*cold*	*ways*	kinds	side	best
stand	law	*office*	anymore	find	he'd
sorry	dreams	*knows*	*running*	you'd	*awful*
realized	face	weren't	set	*concerned*	*inside*
turn	*lying*	*early*	live	number	*recall*
position	*playing*	you'll	intercourse	general	scared
worked	*possible*	walked	*hate*	heard	sudden
fight	putting	*experience*	tired	attitude	afternoon
nervous	*under*	walking	several	*attention*	*tuesday*
immediately	*taken*	*worried*	*began*	o'clock	small
instance	hospital	months	god	living	sunday
wednesday	*explain*	forget	*front*	summer	accept
enjoy	line	outside	run	session	dinner
world	beginning	*liked*	*story*	*eat*	mine
crying	*turned*	act	*wait*	ahead	*apartment*
hope	mood	*behind*	*listen*	unless	woke
walk	guilt	*therefore*	free	struck	books
pick	*men*	month	*ready*	write	*glad*
street	building	using	aren't	conversation	*order*
handle	*buy*	*decision*	looks	she'd	worse
jewish	depressed	fun	terribly	tonight	hair
miserable	silly	black	date	leaving	move
feels	*given*	*interest*	meeting	towards	*lots*
teacher	*fairly*	*older*	reminds	*train*	amount
sat	stupid	view	bother	horrible	period
drive	driving	fighting	*happening*	*knowing*	pleasure
stopping	stopped	anxious	assume	*example*	fall
absolutely	lie	whenever	*evening*	*possibly*	*attractive*
earlier	*brother*	conflict	further	*guys*	it'll

possibility	*reality*	spend	*waiting*	appointment	middle
hit	*itself*	*area*	boys	*died*	excuse
questions	*vacation*	*whom*	obvious	sound	*store*
mother's	beautiful	became	discuss	everyone	*react*
busy	*calling*	*dead*	noticed	*partly*	speak
chance	*clearly*	he'll	role	smoking	*process*
opposite	*physical*	*starting*	stomach	dirty	takes
top	changed	ended	hostility	mouth	nose
remembered	till	we'd	*anyone*	behavior	bye
across	damn	definitely	easy	*extent*	*hold*
please	*relation*	weird	clothes	*death*	*full*
ones	responsibility	*treatment*	father's	lived	*lose*
suddenly	*understanding*	direction	extremely	recognize	she'll
discussed	eyes	jealous	*lost*	pattern	perfectly
uncle	*admit*	bought	step	*study*	*bothered*
level	cry	*express*	friendly	good-by	odd
party	specific	*straight*	totally	*trust*	*attack*
consider	frightened	*letter*	quit	short	*statement*
body	consciously	daddy	easier	embarrassed	game
present	successful	*surprised*	trip	*wall*	*worth*
hasn't	husband	meet	moved	patient	ridiculous
test	*within*	younger	fit	*known*	loved
weak	bathroom	blue	fault	*herself*	learn
rid	stayed	unhappy	agree	*association*	deep
recently	sorts	*two*	*watching*	*writing*	avoid
human	keeps	*mention*	*relations*	annoyed	dark
grandmother	*issue*	marriage	notice	quality	schedule
bothers	break	die	emotional	*follow*	*ideas*
somewhat	they'll	confused	effort	enjoyed	incident
necessarily	piece	smoke	warm	becomes	there's
ran	upon	continue	emotionally	keeping	movie
tense	themselves	tied	choice	*constantly*	describe
impression	*lately*	picked	purpose	*state*	tremendous

voice	*water*	*caught*	discussion	fell	mostly
table	*teach*	*willing*	air	aunt	*conscience*
frustated	push	quiet	tape	*throw*	*truth*
confidence	main	percent	*physically*	*prove*	*town*
bus	*country*	doubt	emotions	gives	likes
shower	*social*	wondering	*becoming*	*company*	competition
library	*listening*	*logical*	machine	*miss*	*sexually*
wrote	basically	*bothering*	chair	*eating*	fears
similar	*thank*	wear	bringing	disappointed	dreamt
honest	*mixed*	response	sad	send	*twice*
christmas	easily	finding	food	forgotten	needed
pregnant	*pressure*	*reasonable*	*underneath*	wearing	degree
heart	hoping	*marry*	needs	pants	patients
quickly	*showed*	speaking	*whereas*	acting	aggressive
brings	charge	correct	critical	cutting	*decisions*
figured	*kill*	*missing*	moving	*places*	pleasant
stuck	various	age	*associate*	finished	*lack*
pictures	someplace	stage	*staying*	*written*	besides
escape	expected	*expression*	*lead*	*learned*	lives
they'd	usual	*anywhere*	ashamed	basis	*calls*
clean	*element*	fellow	*lady*	opposed	passive
research	respect	slept	suit	surface	*worrying*
associations	changing	*city*	described	*fair*	forgot
heck	near	*relate*	shut	simple	situations
student	asleep	childhood	*dressed*	memory.	*showing*
aside	beyond	circumstances	convinced	*excitement*	forward
losing	medical	record	reminded	satisfaction	selfish
significant	sometime	special	*studying*	tension	tremendously
carry	*complete*	doubts	*feet*	generally	masculine
positive	*shows*	*son*	stick	*understood*	unusual
adult	bedroom	birthday	blame	discussing	fat
immediate	intense	major	*reactions*	sessions	smart
born	box	capable	*cause*	*caused*	enjoying

killed	laughing	poor	*terrific*	active	*cannot*
compulsive	direct	doctors	draw	drove	female
frustration	inferior	*letting*	*mental*	power	regard
remind	*tough*	upsetting	accepted	*actual*	*bunch*
emotion	*force*	*hated*	indeed	inner	lawyer
natural	*plans*	quarter	*red*	touch	upstairs
assumed	*character*	*concern*	corner	dare	*essentially*
male	*missed*	*nature*	occasionally	*otherwise*	painful
project	*reacting*	*split*	spoke	theory	*whose*
cases	directly	disgusted	*expressed*	finger	*hall*
hurting	pain	personal	*plus*	*progress*	regular
tie	unpleasant	article	*comment*	commitment	*differently*
fits	hi	*parts*	*practically*	shall	urge
catholic	*context*	dangerous	*downstairs*	*drink*	hassle
hiding	ice	*liking*	opinion	personality	*remark*
roommate	sequence	significance	stronger	strongly	tells
analyze	*apart*	*approach*	argument	block	blood
coffee	*considered*	*exciting*	*helping*	israel	jews
lunch	mess	nasty	overcome	relationships	*resent*
treat	*wished*	afford	*catch*	*contact*	*difficulty*
grow	hardly	*heavy*	*information*	lucky	naked
neat	*negative*	one's	perfect	public	pushing
religion	seriously	they've	*unable*	wake	*window*
awfully	ball	basic	camp	elevator	focus
july	onto	paid	passed	piano	plane
responsible	single	*spring*	*stood*	tight	wishes
based	breathing	build	changes	cigarette	confident
defense	dislike	*downtown*	extreme	held	mainly
opening	opportunity	peculiar	quick	riding	shape
struggle	suggested	talks	*teacher*	*turning*	*turns*
ate	bigger	broke	*center*	colored	*complicated*
convince	daughter	developed	*eventually*	expensive	frustrating
harder	kidding	kitchen	lousy	mail	medicine

phrase	*planning*	program	screaming	sensitive	slow
third	advantage	success	attempt	blow	*carried*

Appendix G. Sources for Touch-Tone Decoders

Devices for Touch-Tone decoding

Standalone devices:

DECtalk - A high-quality speech synthesizer. Manufactured by Digital Equipment Corporation.

Teleport 300 - 300 baud modem. Manufactured by TelTone Corporation.

PC-Compatible devices:

Communicard - Speech synthesizer, digitized speech. Manufactured by Digital Pathways, Inc.

Magnum 10 - Multi-function card, modem/memory/clock. Manufactured by American High Tech, Inc.

Modem 300 / Modem 1200 - Modems with auxiliary ring indicator. Manufactured by Techmar Corporation.

Appendix H. Program Listing

```
#include <stdio.h>

#define BOOL            int
#define CR              0x0D
#define EQUALS          0
#define NULL            0
#define FALSE           0
#define TRUE            1
#define FIRSTGREATER    1
#define FIRSTLESS       -1
#define LF              0x0A
#define CLIP            6
#define MAXSTRLEN       25
#define MAXWORDSIZE     20
#define HTSIZE          1103
#define PRINTER         "prn"
#define VOID            int

extern char         *alloc();
extern char         *strcpy();
extern char         *Copy();

struct WordType{
     char *Letters;
     char *Digits;
} HashTable[HTSIZE];

typedef struct {
     char Seq[25];
     long Score;
}MaybeWords;

MaybeWords Posf[3*CLIP];
MaybeWords Posb[3*CLIP];
int  Tgm[28][28][28];
int  LowLimit;
char *location;


main()
{

     static char datname[16] = "a:trigram.dat";
     long      LookWord();
     char      CurNums[MAXSTRLEN];
     char      CurWord[MAXSTRLEN];
     int       EndOfFile = FALSE;
     int       EndOfHT = FALSE;
     long      HTIndex;
     FILE      *DataFile;
     FILE      *InFile;
     FILE      *ListFile;
     int       *ptr, i, j, k;
```

```
    int       WordLen;

    location = alloc(12000);

    for (i = 0; i < HTSIZE; i++) {
         HashTable[i].Letters = NULL;
         HashTable[i].Digits = NULL;
    }

    if (!(InFile = fopen("a:diction.txt","r"))){
         printf("\nThe file diction.txt is not on the disk\n");
         exit(0);
    }

    while (EndOfFile == FALSE && EndOfHT == FALSE){
         EndOfFile = GetNextWord(InFile,CurWord,CurNums);
         if (!EndOfFile)
              EndOfHT = InsertWord(CurWord,CurNums);
    }
    fclose(InFile);

    if (!(DataFile = fopen(datname,"rb"))){
         printf("\nDatafile %s doesn't exist.",datname);
         printf("\nSee what you can do about it.");
         exit(0);
    }

    ptr=&Tgm[0][0][0];
    while((i=getw(DataFile)) != EOF){
         *ptr = i;
         ++ptr;
    }

    fclose(DataFile);
    printf("Enter value for the trigram cutoff--");
    scanf("%d",&LowLimit);

    while (TRUE){
         scanf("%s",CurNums);
         GenWords(CurNums);
         HTIndex = LookWord(CurNums);
         if (HTIndex >= 0){
              printf("%s\n",HashTable[HTIndex].Letters);
         }
         else
              printf("Not a word in the list.\n");
    }
} /* MAIN */

/*****************************************************************************/

BOOL GetNextWord(WhichFile,Word,Numbers)
FILE      *WhichFile;
```

```
char      *Numbers;
char      *Word;
{
     int  CurChar;

     while ((CurChar = GetCStripped(WhichFile)) != EOF)
          if (isalpha(CurChar))
               break;

     do {
          if (NotValid(CurChar))
               break;
          else{
               *Word++ = toupper(CurChar);
               *Numbers++ = ToDigit(CurChar);
          }
     }

     while ((CurChar = GetCStripped(WhichFile)) != EOF);

     *Word = '\0';
     *Numbers = '\0';

     return ((CurChar == EOF) ? TRUE : FALSE);
}

/*****************************************************************************/

BOOL InsertWord(Word,Numbers)
char      *Numbers;
char      *Word;
{
     long      Hash();
     long      HTIndex;


HTIndex = Hash(Numbers);

if (HTIndex == -1)
     return(TRUE);

if (HashTable[HTIndex].Digits == NULL){
     HashTable[HTIndex].Letters = Copy(Word);
     HashTable[HTIndex].Digits = Copy(Numbers);
}
else
     /* DO NOTHING */;
return(FALSE);
}

/*****************************************************************************/

VOID LowerString(Value)
```

```
char      *Value;
{
     for ( ; *Value != '\0'; Value++)
          *Value = tolower(*Value);
}

/*****************************************************************************/

int ToDigit(CurChar)
int       CurChar;
{
     int  Digit;

     CurChar = toupper(CurChar);

     switch(CurChar){
          case 'A':
          case 'B':
          case 'C':
               Digit = '2';
               break;
          case 'D':
          case 'E':
          case 'F':
               Digit = '3';
               break;
          case 'G':
          case 'H':
          case 'I':
               Digit = '4';
               break;
          case 'J':
          case 'K':
          case 'L':
               Digit = '5';
               break;
          case 'M':
          case 'N':
          case 'O':
               Digit = '6';
               break;
          case 'P':
          case 'R':
          case 'S':
               Digit = '7 ;
               break;
          case 'T':
          case 'U':
          case 'V':
               Digit = '8';
               break;
          case 'W':
```

```
          case 'X':
          case 'Y':
               Digit = '9';
               break;
          case 'Q':
          case 'Z':
          case '\047':
          case '.':
          case '-':
               Digit = '1';
               break;
          default:
               printf("ToDigit barfed on %c",CurChar);
               return('$');
     }
     return(Digit);
}

/*****************************************************************************/

BOOL NotValid(CurChar)
int       CurChar;
{
     if (isalpha(CurChar))
          return(FALSE);
     if (ispunct(CurChar))
          return(FALSE);
     else
          return(TRUE);
}

/*****************************************************************************/

long LookWord(Numbers)
char      *Numbers;
{
     long      Locate();
     long      HTIndex;


HTIndex = Locate(Numbers);

return(HTIndex);

if (HTIndex < 0)
     return(HTIndex);

if (strcmp(HashTable[HTIndex].Digits,Numbers)){
     return(HTIndex);
}
else
     return(-2);
}
```

```
/*****************************************************************************/

long Locate(Dig_string)
char *Dig_string;
{
     long      HTIndex;
     long      IdLen;
     long      InitHTIndex;
     long      ProbeCounter = 0;
     long      GenerateNewIndex();
     long      TransformId();

     if (!(IdLen = strlen(Dig_string)))
          printf("Hash: Dig_string of no length\n");
     HTIndex = InitHTIndex = TransformId(Dig_string);

     if (*HashTable[HTIndex].Digits == NULL)
          return(-3);
     else
          if (strcmp(Dig_string,HashTable[HTIndex].Digits) == EQUALS)
               return(HTIndex);
          else
               for (ProbeCounter=0;ProbeCounter < (HTSIZE/2);ProbeCounter++){
                    HTIndex = GenerateNewIndex(InitHTIndex,ProbeCounter);
                    if (*HashTable[HTIndex].Digits == NULL)
                         return(-4);
                    else if (strcmp(Dig_string,HashTable[HTIndex].Digits)==EQUALS)
                         return(HTIndex);
               }
     if (ProbeCounter >= (HTSIZE/2))
          return(-5);
     return(-6);
}

/*****************************************************************************/

int GetCStripped(WhichFile)
FILE *WhichFile;
{
int       Temp;

     if ((Temp = getc(WhichFile)) != EOF)
          return(Temp & 0x7f);
     return(EOF);
}

/*****************************************************************************/

char *Copy(OldString)
char *OldString;
{
     char *NewString;
```

```
    int len;

    len = strlen(OldString);
    NewString = location;
    strcpy(location,OldString);
    location += (len+1);
    return(NewString);
}

/*****************************************************************************/

long TransformId(Dig_string)
char Dig_string[];
{
    long     Term;
    long     WordIndex;

for (Term=0, WordIndex = strlen(Dig_string) - 1; WordIndex > -1; WordIndex--)
    Term = (257 * Term) + Dig_string[WordIndex];
Term = (Term < 0) ? -Term : Term;
return(Term % HTSIZE);
}

/*****************************************************************************/

long GenerateNewIndex(OriginalKey,ProbeNumber)
long     OriginalKey;
long     ProbeNumber;
{
    return((OriginalKey + ProbeNumber * ProbeNumber) % HTSIZE);
}

/*****************************************************************************/

long Hash(Dig_string)
char *Dig_string;
{
    long     HTIndex;
    long     IdLen;
    long     InitHTIndex;
    long     ProbeCounter = 0;
    long     GenerateNewIndex();
    long     TransformId();

    if (!(IdLen = strlen(Dig_string)))
        printf("Hash: Dig_string of no length\n");
    HTIndex = InitHTIndex = TransformId(Dig_string);

    if (*HashTable[HTIndex].Digits == NULL)
        /*NULL*/;
    else
        if (strcmp(Dig_string,HashTable[HTIndex].Digits) == EQUALS)
            /*ALREADY THERE*/;
```

```
          else
               for (ProbeCounter=0;ProbeCounter < (HTSIZE/2);ProbeCounter++){
                    HTIndex = GenerateNewIndex(InitHTIndex,ProbeCounter);
                    if (!HashTable[HTIndex].Digits == NULL)
                         break;
                    else if (strcmp(Dig_string,HashTable[HTIndex].Digits) == EQUALS)
                         break;
               }
     if (ProbeCounter >= (HTSIZE/2))
          return(-1);
     return(HTIndex);
}

/*****************************************************************************/

VOID Zero_list()

{
     int i;

     for(i=0; i<3*CLIP; i++){
          Posf[i].Seq[0] = '\0';
          Posf[i].Score = 0;
          Posb[i].Seq[0] = '\0';
          Posb[i].Score = 0;
     }
}

/*****************************************************************************/

VOID GenWords(CurNums)
char *CurNums;
{

     int len, i, j, k;
     int digit[32];
     long tritemp;
     char digchar[2];
     static int Letters[12][3] ={
          {92, 92, 92},
          {81, 90, 92},
          {65, 66, 67},
          {68, 69, 70},
          {71, 72, 73},
          {74, 75, 76},
          {77, 78, 79},
          {80, 82, 83},
          {84, 85, 86},
          {87, 88, 89},
          {91, 91, 91},
          {92, 92, 92}
     };
```

```
len = strlen(CurNums);
if (len < 2)
     return;

for(i=0; i<32; i++)
     digit[i]=0;

for(i=0; i<len; i++){
     digchar[0] = *CurNums;
     digchar[1] = '\0';
     digit[i] = atoi(digchar);
     CurNums++;
}

for(j=0; j<3; j++){
     for(k=0; k<3; k++){
          Posf[3*j+k].Seq[0] = Letters[digit[0]][j];
          Posf[3*j+k].Seq[1] = Letters[digit[1]][k];
          Posf[3*j+k].Score=Tgm[26][Letters[digit[0]][j]-65]
                                   [Letters[digit[1]][k]-65];
          Posf[3*j+k].Seq[2] = '\0';
     }
}

for(i=9; i<3*CLIP; i++)
     Posf[i].Score = 0;

SortList(&Posf[0]);

for(i=2; i < len; i++){
     for(j=0; j<CLIP; j++){
          for(k=0; k<3; k++){
               Posf[CLIP*k+j].Seq[i] = Letters[digit[i]][k];
               tritemp = Tgm[Posf[CLIP*k+j].Seq[i-2]-65][Posf[CLIP*k+j].Seq[i-1]-65][Letters[digit[i]][k]-65];
               if(tritemp<=LowLimit)
                    Posf[CLIP*k+j].Score=0;
               else
                    Posf[CLIP*k+j].Score += tritemp;
               Posf[CLIP*k+j].Seq[i+1] = '\0';
          }
     }
     SortList(&Posf[0]);
}

for(i=0; i<CLIP; i++){
     Posf[i].Score +=Tgm[Posf[i].Seq[len-2]-65][Posf[i].Seq[len-1]-65][26];
     Posf[i+CLIP].Score = Posf[i+2*CLIP].Score = 0;
}

SortList(&Posf[0]);

for(i=0; i<3*CLIP; i++){
     for(j=0; j<len; j++){
```

```
                Posb[i].Seq[j] = '*';
            }
        }

        for(j=0; j<3; j++){
            for(k=0; k<3; k++){
                Posb[3*j+k].Seq[len-1]=Letters[digit[len-1]][j];
                Posb[3*j+k].Seq[len-2]=Letters[digit[len-2]][k];
                Posb[3*j+k].Score=Tgm[Letters[digit[len-2]][k]-65]
                                     [Letters[digit[len-1]][j]-65][26];
                Posb[3*j+k].Seq[len] = '\0';
            }
        }

        for(i=9; i<3*CLIP; i++)
            Posb[i].Score = 0;

        SortList(&Posb[0]);

        for(i=2; i < len; i++){
            for(j=0; j<CLIP; j++){
                for(k=0; k<3; k++){
                    Posb[CLIP*k+j].Seq[len-1-i]=Letters[digit[len-1-i]][k];
                    tritemp = Tgm[Letters[digit[len-1-1]][k]-65][Posb[CLIP*k+j].Seq[len-1]-65][Posb[CLIP*k+j].Seq[len-1+1]-65];
                    if(tritemp<=0)
                        Posb[CLIP*k+j].Score=0;
                    else
                        Posb[CLIP*k+j].Score += tritemp;
                }
            }
            SortList(&Posb[0]);
            }

        for(i=0; i<CLIP; i++){
            Posb[i].Score +=Tgm[26][Posb[i].Seq[0]-65][Posb[i].Seq[1]-65];
            Posb[i+CLIP].Score = Posb[i+2*CLIP].Score = 0;
        }

        SortList(&Posb[0]);
        SortMerge();

        for(i=0; i<CLIP; i++)
            printf(" %s %ld",Posf[i].Seq,Posf[i].Score);
        printf("\n");


} /***** GENWORDS *****/

/*****************************************************************************/

VOID SortList(list)
MaybeWords list[];
{
```

```
    int i, j, k;
    long Tempscore;
    char Tempseq[32];

    for(i=0; i<CLIP; i++){
        for(j=i+1; j<3*CLIP; j++){
            if(list[j].Score > list[i].Score){
                Tempscore = list[i].Score;
                strcpy(Tempseq,list[i].Seq);
                list[i].Score = list[j].Score;
                strcpy(list[i].Seq,list[j].Seq);
                list[j].Score = Tempscore;
                strcpy(list[j].Seq,Tempseq);
            }
        }
    }
    for(i=0; i<CLIP; i++){
        list[CLIP+i].Score=list[CLIP*2+i].Score=list[i].Score;
        strcpy(list[CLIP+i].Seq,list[i].Seq);
        strcpy(list[CLIP*2+i].Seq,list[i].Seq);
    }
} /*** SORTLIST ***/

/*****************************************************************************/

VOID SortMerge()

{

    int i, j, k;
    long Tempscore;
    char Tempseq[32];

    for(i=0; i<CLIP; i++){
        for(j=0; j<CLIP; j++){
            if(!strcmp(Posf[i].Seq,Posb[j].Seq))
                Posb[j].Score = 0;
        }
    }
    for(i=0; i<CLIP; i++){
        strcpy(Posf[CLIP+i].Seq,Posb[i].Seq);
        Posf[CLIP+i].Score = Posb[i].Score;
    }
    for(i=0; i<CLIP; i++){
        for(j=i+1; j<2*CLIP; j++){
            if(Posf[j].Score > Posf[i].Score){
                Tempscore = Posf[i].Score;
                strcpy(Tempseq,Posf[i].Seq);
                Posf[i].Score = Posf[j].Score;
                strcpy(Posf[i].Seq,Posf[j].Seq);
                Posf[j].Score = Tempscore;
                strcpy(Posf[j].Seq,Tempseq);
            }
```

```
        }
    }
} /*** SORTMERGE ***/

/*****************************************************************************/
```