Computer detection of misspelled words

Symonds, Martha

Report (National Research Council of Canada. Radio and Electrical Engineering Division : ERB), 1970-08

ERB-843

UNCLASSIFIED

NATIONAL RESEARCH COUNCIL OF CANADA

RADIO AND ELECTRICAL ENGINEERING DIVISION

COMPUTER DETECTION OF MISSPELLED WORDS

- MARTHA SYMONDS -

OTTAWA

AUGUST 1970

ABSTRACT

Various algorithms for computer detection of misspellings have been proposed. Techniques described by three authors have been evaluated and compared to a method based on pronunciation rules, in which a set of character strings is automatically produced from an English word. The size of the set is variable; each member represents a possible 'pronunciation' of the word. For any two words, their respective sets of 'pronunciations' can be compared, and, on this basis, it can be determined if one word is a misspelling of the other. This method produced the lowest number of errors.

CONTENTS

COMPUTER DETECTION OF MISSPELLED WORDS

– Martha Symonds –

Introduction

Systems involving the processing of natural language text by computer in general require some method of detecting and correcting errors in the input words. At the National Research Council, a study is under way of the application of computers in instructional systems. Early in this study it became apparent that a facility for the detection of spelling errors was necessary in such systems, particularly when the primary input device was a keyboard. Without this facility, it can become quite tedious for the student to have minor spelling mistakes rejected as errors, and it is difficult for the system to obtain a true picture of the student's progress. The possibility of entering all common misspellings of answers is discarded as unworkable.

Input errors fall into two classes. The first of these includes errors due to random substitution, addition, or deletion of letters resulting from noise, or typing errors. The second is made up of *human* spelling errors as opposed to *equipment* spelling errors. An effective method of detecting spelling errors must be capable of handling both classes of errors.

Blair [1] has proposed a method of error detection using abbreviation of the words. For any misspelling, the four letters least likely to be in error form the abbreviation. Each letter of the word is assigned a code number based on the letter itself and its position in the word. The letters with the highest code numbers are successively deleted, until a four-letter abbreviation is formed. This abbreviation is compared with the similarly generated abbreviations of correctly spelled words; matching abbreviations indicate the correct spelling. The code numbers used by Blair are based on the likelihood of human error and, therefore, this abbreviation method is not designed to correct machine-generated misspellings.

Damerau [2] has proposed that most spelling errors – whether human or not – consist of a single instance of one of the following:

1. Substitution of a letter
2. Deletion of a letter
3. Insertion of a letter
4. Transposition of two adjacent letters

A misspelling differing from a dictionary word in one of these respects is judged a misspelling of that word.

Besides testing the methods of Blair and Damerau among others, Alberga [3] proposes several methods based on the construction of coincidence matrices. For a misspelling of length m and a correctly spelled word of length n, an $m \times n$ matrix m_{ij} is formed such that:

$$m_{ij} = 1 \text{ if } i\text{-th letter of misspelling matches } j\text{-th letter of correctly spelled word}$$
$$= 0 \text{ otherwise .}$$

Various weighting and selection routines can be applied to this matrix. Most of the weighting procedures make use of the fact that 1's near the zero diagonal are more likely to be significant than those further away. Another of the procedures uses the immediately surrounding elements of the matrix to weight each element.

The selection routines ensure that in each row and column of the coincidence matrix, there is, at most, one non-zero element. By applying one of these procedures, no letter of either word will be considered a match for more than one letter of the other word.

A similarity function applied to the resulting matrix produces a measure of the probability of a match between the two words. Provided that this measure exceeds an empirically chosen value, the misspelling is considered a match for the correctly spelled word.

Alberga also mentions a method based on word pronunciation. This program was not described in much detail and was discarded as unusable.

Pronunciation Method

With a method based solely on the pronunciation of words, the detection of spelling errors due to random substitution, addition, or deletion of letters (e.g., typing errors) is not possible. Such a program is expected to detect only human spelling errors.

It is hypothesized that a large number of misspellings are phonetically correct. Furthermore, the consonants of a misspelled word are more likely than the vowels to be phonetically correct. On this assumption, the pronunciation of the vowels in a word is not considered in the program.

From each input word, the program produces a set of 'pronunciations' in the following manner. The word is scanned from left to right, and tested for the occurrence of consonants and some common digrams and trigrams (e.g., ch, sh, sci, tio). For each instance a code is concatenated to the end of the character string representing the pronunciation of the consonants [4]. (This string is initialized as a character string of zero length).

The number of possible pronunciations depends on what letters occur in the word. Some have only one common pronunciation (e.g., b) and others have up to three (e.g., ch as in ma*ch*ine, *ch*air, *ch*orus). Thus there may be up to three different codes which could be added to the string. Because of the irregularity of English pronunciation there is no way of automatically choosing the correct pronunciation. Thus each of the possible codes is concatenated in turn to the end of the string in effect doubling or tripling the number of pronunciations.

The consonants, digrams, and trigrams considered in this program and the codes assigned to them are listed in Table 1.

Table 1

String considered	Number of pronunciations	Possible codes	Examples	Restrictions
b	1	b	*b*ut	
ch	3	s2	ma*ch*ine	
		c2	*ch*air	
		k	*ch*orus	
ck	1	k	ba*ck*	
cq	1	q	a*cq*uire	
c	2	s	*c*ity	
		k	*c*an	
dg	2	j	ba*dg*e	
		g		
d	1	d	*d*og	
f	1	f	*f*ill	
gn	1	n	*gn*aw	
ght	2	t	slau*ght*er	
		ft	lau*ght*er	
gh	2	f	lau*gh*	
		g	*gh*ost	
g	2	j	*g*em	
		g	*g*um	
h	1	h	*h*at	
j	1	j	*j*oke	
kn	1	n	*kn*ot	
k	1	k	*k*eep	
l	1	l	*l*ate	
m	1	m	*m*an	
n	1	n	*n*od	
ph	1	f	*ph*antom	
p	1	p	*p*en	

cont'd

Table 1 (continued)

String considered	Number of pronunciations	Possible codes	Examples	Restrictions
q	1	q	*qu*een	
r	1	r	*r*at	
rh	1	r	*rh*ubarb	
sci or sce	2	s2	con*sci*ence	
		s	*sci*ence	
sh	1	s2	*sh*e	
s	2	z	eas*y*	
		s	*s*afe	
tch	3	s2		
		c2	la*tch*	
		k		
tio	2	c2	ques*tio*n	
		s3	na*tio*n	
th	1	t2	*th*e	
t	1	t	*t*ake	
v	1	v	*v*an	
wh	1	w	*wh*en	
w	1	w	*w*ay	'w' not coded if it is last letter in word
x	2	ks	ve*x*	
		gz	e*x*ist	
y	1	y	*y*et, mone*y*	'y' not coded unless it is first or last letter of word
z	1	z	*z*oo	

Adjacent occurrences of the same consonant in a word are considered equivalent to a single occurrence of that consonant. Some instances of consonants are ignored (e.g., 'w' when it is the final letter, 'y' except when it is the initial or final letter).

Some examples of input words and the character strings produced from them by the program are shown in Table 2.

Table 2

WORD										
1.	FIR	' '	→	'F'	→	'FR'				
2.	BORDER	' '	→	'B'	→	'BR'	→	'BRD'	→	'BRDR'
3.	DOCTOR	' '	→	'D'	→	'DS'	→	'DST'	→	'DSTR'
					→	'DK'	→	'DKT'	→	'DKTR'
4.	STRAIGHT	' '	→	'Z'	→	'ZT'	→	'ZTR'	→	'ZTRT'
									→	'ZTRFT'
			→	'S'	→	'ST'	→	'STR'	→	'STRT'
									→	'STRFT'
5.	ACROSS	' '	→	'S'	→	'SR'	→	'SRZ'		
							→	'SRS'		
			→	'K'	→	'KR'	→	'KRZ'		
							→	'KRS'		

For two words, their respective sets of 'pronunciations' are compared. If any two members of the sets match, one word is then considered a misspelling of the other.

In practice certain restrictions are imposed on this comparison process. The 'pronunciations' of two words are not compared if the two words differ in length by more than two characters. The probability that two words of greatly differing lengths are differently spelled versions of the same word is low and, to save computer time, such words are not compared.

Also, two words are not compared if one begins with a vowel and the other with a consonant. This step was introduced on the basis of preliminary results – and it appears logical that very few misspellings would involve the substitution of an initial vowel for a consonant, and vice versa.

Each word does not give rise to a unique set of character strings – different words may share one or more of the same possible pronunciations. Thus it is quite possible for a word to be identified by the program as a misspelling of an entirely different word. The frequent occurrence of this kind of error would greatly reduce the usefulness of the program. However, results indicate that this does not happen. In the application to computer aided teaching, the number of acceptable answers will be relatively small and the chances that a response will be incorrect and have a character string code in common with one of the correct answers are not too great.

The other type of error that may occur is the failure to recognize a misspelling as such. Distinguishing between misspellings and actual errors is naturally somewhat arbitrary. The more obvious misspellings do, however, tend to be phonetically correct, at least in the consonants, and are thus recognized by the program.

In some cases misspellings are not recognized. This occurs most frequently in words with silent consonants. Practically every consonant is silent in some words and there seem to be no strict rules determining whether or not a consonant is pronounced.

One way to solve this would be to include as a possible pronunciation of each consonant a code of zero length. However, this would completely destroy the effectiveness of the program as each word would have as one of its possible pronunciations a character string of zero length. The program would thus recognize each word as a misspelling of every other word. It is preferable to miss some obvious misspellings (e.g., Febuary for February) than to introduce so great a source of error. Some provision is made, however, for the more widespread instances of silent consonants (see Table 1, e.g., 'gn' and 'kn').

A more promising solution to the problem of identifying misspellings of words with silent consonants is to use the method of error detection proposed by Damerau as a preliminary step. This method is particularly well suited to picking out misspellings formed by omitting a single silent consonant.

Changes in the pronunciation program so that it may identify as many misspellings as possible increase the likelihood that a word will be identified as a misspelling of the wrong word. Similarly, any modifications used to decrease wrong identifications will probably result in more misspellings being missed . Thus attempts to eliminate one source of error increase the chances that the other kind of error will occur. A compromise must be found and is best arrived at empirically.

Results

The pronunciation program, and the programs proposed by Blair and Damerau, were written initially in Fortran. These programs were later written in PL/1 to take advantage of the string manipulation facilities of that language.

The programs were run on an IBM 360/50 computer. Most testing was done on the PL/1 version of the pronunciation program which includes some minor revisions not present in the Fortran program. In one test a list of 500 correctly spelled, randomly chosen words was used. Each word in turn was tested to see if it was a misspelling of any other. Of the 500 words, 18 pairs have at least one 'pronunciation' in common and are mistakenly identified as misspellings of each other. Therefore, for 93% of the words, incorrect identifications do not occur.

In another test, 320 misspellings of about 100 words were obtained from the results of a grade 5 spelling test. When these misspellings were compared with each of the correctly spelled words, no match was found for the misspellings in 108 cases. For the same set of words, Blair's abbreviation method failed to recognize 201 words and one of the better methods proposed by Alberga failed to recognize 162 words.

Two pairs of the correctly spelled words in this test have 'pronunciations' in common and misspellings of one word were identified as misspellings of both words. This happened four times; thus in the test, almost 99% of the words were not identified as misspellings of the wrong word. In 21 cases, the misspellings produced different correctly spelled English words. However, these new words were not in the list of correct words and were treated as misspellings like the rest.

It does not seem reasonable to include as failures of the program, words so badly misspelled that they cannot be recognized at all. Therefore, the 108 words that were not recognized by the pronunciation technique were divided into 5 lists. Twenty-five people were given a copy of one of the lists and asked to guess the correct words. Of the 108 misspellings, 46 were recognized by all the people in the test and 70 were recognized by at least 75% of the people. Thus at least 38 misspellings can be discarded, meaning that at least 78% of these words were properly identified by the pronunciation technique.

In a comparative test of all the methods, 566 correctly spelled words and 108 misspellings of these words were used. The number of errors, time of execution and storage requirements are compared in Table 3. In the test combining the pronunciation and single error methods, misspellings are first compared with each dictionary word by the single error program. If no match is found, the pronunciation routine is applied successively for the misspelling and each dictionary word.

Table 3

Program	Language	% of Words Correctly and Uniquely Identified	Execution Time (Sec/Number of Misspellings)	Program Length (Bytes)
PRON	PL/1	90	3.7	38K
DAMR	PL/1	79	.6	25K
	Fortran (H compiler)	79	.6	78K
BLAIR	PL/1	77	7.7	27K
CONTEX + SBYC + PAIRS	Fortran (H compiler)	84	14.8	81K
PRON + DAMR combined	PL/1	94	1.8	40K

Dictionary Length = 566 words
PRON = pronunciation routine
DAMR = single error routine
BLAIR = abbreviation routine

CONTEX + SBYC + PAIRS = one of Alberga's routines where:
CONTEX = weighting procedure
SBYC = selection procedure
PAIRS = similarity function

Conclusions

Blair's abbreviation method has the double disadvantage of the highest error rate and an excessively large execution time. The method proposed by Alberga tested here had a fairly low error rate but the longest execution time of all – about 25 times that of Damerau's method.

The technique with the fastest execution time is that of Damerau. Yet this method is rather restrictive in the type of errors it can detect, and a fairly large number of obvious misspellings are missed.

The method with the lowest error rate is the pronunciation method. The execution time is, however, six times that of Damerau's technique. By using the single error method as a preliminary step to the pronunciation method, the error rate is decreased 4%, and the average execution time per word is halved. This combination of methods is the best solution among those tested for the problem of automatic detection of misspellings.

References

1. Blair, C.R. A program for correcting spelling errors. Inform. Contr., 3: 60–67; 1960.
2. Damerau, F. A technique for computer detection and correction of spelling errors. Comm. A.C.M., 7 (3): 171–176; 1964.
3. Alberga, C.N. String similarity and misspellings. Comm. A.C.M., 10 (5): 302–313; 1967.
4. Bethel, J.P. (Ed.) A guide to pronunciation. *In* Webster's New Collegiate Dictionary, G. & C. Merriam Co., Springfield, Mass., 1960.