Probability Scoring for Spelling Correction

Kenneth W. Church
William A. Gale

AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, N.J., USA

Keywords: automated learning, spelling correction, n-gram language model,

Good Turing estimates.

Abstract

This paper describes a new program, *correct*, which takes words rejected by the Unix® *spell* program, proposes a list of candidate corrections, and sorts them by a probability scores. The probability socres are the novel contribution of this work. They are based on a noisy channel model. It is assumed that the typist knows what words he or she wants to type but some noise is added on the way to the keyboard (in the form of typos and spelling errors). Using a classic Bayesian argument of the kind that is popular in recognition applications, especially speech recognition (Jelinek, 1985), one can often recover the intended correction, *c*, from a typo, *t*, by finding the correction *c* that maximizes $Pr(c)\ Pr(t|c)$. The first factor, $Pr(c)$, is a prior model of word probabilities; the second factor, $Pr(t|c)$, is a model of the noisy channel that accounts for spelling transformations on letter sequences (insertions, deletions, substitutions and reversals). Both sets of probabilities were estimated using data collected from the Associated Press (AP) newswire over 1988 and 1989 as a training set. The AP generates about 1 million words and 500 types per week.

In evaluating the program, we found that human judges were extremely reluctant to cast a vote given only the information available to the program, and that they were much more comfortable when they could see a concordance line or two. The second half of this paper discusses some very simple methods of modeling the context using n-gram statistics. Although n-gram methods are much too simple (compared with much more sophisticated methods used in AI and natural

language processing), we have found that even these very simple methods illustrate some very interesting estimation problems that will almost certainly come up when we consider more sophisticated models of contexts. The problem is how to estimate the probability of a context that we haven't seen. We compare several estimation techniques and find that some are useless or worse than useless. Fortunately, we have found that the Good-Turing method provides an estimate of contextual probabilities that produces a significant improvement in program performance. Context is helpful in this application, but only if it is estimated very carefully.

At this point, we have a number of different knowledge sources: the prior, the channel and the context, and there will certainly be more in the future. In general, performance will be improved as more and more knowledge sources are added to the system, as long as each additional knowledge source provides some new (independent) information. As we shall see, it is important to think more carefully about combination rules, especially when there are a large number of different knowledge sources.

1. The Problem

The *correct* program reads a list of misspelled words from the input stream (*stdin*) and prints a set of candidate corrections for each word on the output stream (*stdout*). *Correct* also produces a probability estimate along with each correction (unless there is only one candidate correction). These probabilities distinguish *correct* from previous spelling correctors. The problem of finding just the right way to estimate these probabilities from the various available sources of knowledge poses a technical challenge for both statistics and artificial intelligence. In this paper we have only dealt with a few of the more obvious possibilities.

Here is some sample output produced by the Unix® command, ‘‘spell paper | correct,’’ where *paper* is a text file containing the misspelled words in column 1:

Table 1

Examples of Output of *correct*

Typo	**Corrections**
detered	deterred (100%) metered (0%) petered (0%)
laywer	lawyer (100%) layer (0%) lawer (0%)
negotations	negotiations
notcampaigning	???[1]
progession	progression (94%) procession (4%) profession (2%)
ususally	usually
winky	windy (69%) wink (20%) winks (7%) kinky (2%) wonky (1%) pinky (1%) dinky (0%) winy (0%) inky (0%)

2. Proposing Candidate Corrections

The first stage of *correct* searches a wordlist to find candidate corrections that differ from the input

0. *???* indicates that no correction was found.

typo by a single insertion, deletion, substitution or reversal. The wordlist was collected from five sources, the AP newswire, the Unix® *spell* program, the *Cobuild Dictionary* (Sinclair *et al.*, 1987), the *Collins Dictionary of the English Language* (Hanks *et al*, 1979), and *Roget's International Thesaurus, Fourth Edition* (Chapman, 1977). This first stage produces the following output, given the input typo, *acress*:

Table 2

Example of Candidate Corrections

Typo	**Correction**	**Transformation**			
acress	actress	@	t	2	deletion
acress	cress	a	#	0	insertion
acress	caress	ac	ca	0	reversal
acress	access	r	c	2	substitution
acress	across	e	o	3	substitution
acress	acres	s	#	4	insertion
acress	acres	s	#	5	insertion

The candidate *actress*, for example, can be mistyped as *acress* by deleting the *t* at position 2. The symbols @ and # represent nulls. (The transformations are named from the point of view of the correct word, not the typo.) The typo, *acress*, can arise from *acres* by insertion of *s* after either the fourth or the fifth letter, so *acres* appears twice in the table as a candidate correction. This unusually difficult example was selected to illustrate the four transformations; most examples do not have so many high-scoring possibilities.

In principle, one could implement the transformations very straightforwardly by trying all the possibilities. The insertion operation is actually implemented this way. It requires n dictionary accesses to see if the n^{th} letter in the typo may have been inserted. Unfortunately, the deletion operation is much more expensive; there are 26 letters that might have been deleted in $n+1$ positions. In order to reduce the number of dictionary accesses, the system makes use of a precomputed table of the form:

Table 3

An Example from the Deletion Table

Key	**Correction**	
grea	t	4
gret	a	3
grat	e	2
geat	r	1
reat	g	0

This table maps words missing a letter to corrections. With this table, the system can check for deletions in one table lookup. The table is also useful for checking for substitutions and reversals. Of course, there is a cost in space. The deletion table has about a million entries, approximately ten times as many as the dictionary. The deletion table is stored using heuristic hashing methods very similar to those in *spell*.

3. Scoring

Two versions of *correct* have been studied: one with context and the other without context. We discuss the simpler no-context version first.

Each candidate correction is scored by the Bayesian combination rule $Pr(c)\ Pr(t|c)$, and then normalized by the sum of the scores for all proposed candidates. Care must be taken in estimating the prior because of sparse data problems. It is possible (and even likely) that a proposed correction might not have appeared in the training set. Some methods of estimating the prior would produce undesirable results in this case. For example, the maximum likelihood estimate (MLE) would estimate $Pr(c) = 0$, and consequently, many candidate corrections would be rejected just because they did not happen to appear in the training set (the 1988 AP corpus). We will encounter even more severe forms of the sparse data problem when we consider context.

We will consider three estimation methods that deal with the sparse data problems in three different ways. All of these methods attempt to estimate a set of probabilities, p, from observed frequencies, r. It is assumed that the observed frequencies are generated by a binomial process with N total observations. The estimation methods generate an adjusted frequency r^*, where r^* is a function of r. Once r^* has been determined, then p is estimated as $p \approx r^*/N^*$. $N^* = \sum r^*\, N_r$ where N_r is the frequency of frequency r,[2] assuring that the estimated probabilities add to one. The maximum likelihood estimator (MLE) sets $r^* = r$. The MLE estimate is particularly poor when $r = 0$, since the true probabilities are probability greater than 0 in this case.

Following Box and Tiao (1973), we can assume an uninformative prior and reach a posterior distribution for p. Using the expectation of this distribution amounts to using $r^* = r + 0.5$. We call this the expected likelihood estimate (ELE). The third method is the Good-Turing (GT) method (Good, 1953). This method sets $r^* = (r+1)\ N_{r+1}/N_r$. Unlike the MLE, both ELE and GT estimates assign nonzero probabilities, even when $r = 0$. This is probably a desirable property.

The prior, $Pr(c)$, estimated by the ELE, is $(freq(c) + 0.5)/(N + V/2)$, where $freq(c)$ is the number of times that the proposed correction c appears in the 1988 AP corpus ($N = 44$ million words, and the vocabulary size, V, is 112964). Although we have discussed problems with the ELE elsewhere (Gale and Church, 1990), it will be used here.

The channel probabilities, $Pr(t|c)$, are computed from four confusion matrices (presented in the appendix so as to encourage replication of this work): (1) $del[x,y]$, the number of times that the characters *xy* (in the correct word) were typed as *x* in the training set, (2), $add[x,y]$, the number of times that *x* was typed as *xy*, (3) $sub[x,y]$, the number of times that *y* was typed as *x*, and (4) $rev[x,y]$, the number of times that *xy* was typed as *yx*. Probabilities are estimated from these matrices by dividing by $chars[x,y]$ or $chars[x]$, the number of times that *xy* and *x* appeared in the training set, respectively. As the following formula shows, these four confusion matrices are normalized by $chars[x,y]$ and $chars[x]$, the number of times that *xy* and *x* appeared in the training

2. More formally, N_r is the number of types that have frequency r.

set, respectively. These matrices are not included in the appendix since they can be easily replicated.

$$Pr(t|c) \approx \begin{cases} del[c_{p-1}, c_p]/chars[c_{p-1}, c_p] & \text{if deletion} \\ add[c_{p-1}, t_p]/chars[c_{p-1}] & \text{if insertion} \\ sub[t_p, c_p]/chars[c_p] & \text{if substitution} \\ rev[c_p, c_{p+1}]/chars[c_p, c_{p+1}] & \text{if reversal} \end{cases}$$

In this formula, c_p is the p^{th} character of the proposed correction, c. Likewise, t_p is the p^{th} character of the typo, t. The confusion matrices are computed with a bootstrapping procedure. Initially assume a uniform distribution over the possible confusions. Then run the program over the training set to find corrections for the words that *spell* rejects. Use these corrections to update the confusion matrices, and iterate. The probabilities are estimated using the Good-Turing method (Good, 1953), with the cells of the matrices as the types.

Returning to the *acress* example, the seven proposed transformations are scored by multiplying the prior probability (which is proportional to 0.5 + column 4 in the table below) and the channel probability (column 5) to form a raw score (column 3), which are normalized to produce probabilities (column 2). The final results are: *acres* (45%), *actress* (37%), *across* (18%), *access* (0%), *caress* (0%), *cress* (0%). This example is very hard; in fact, the second choice is probably right, as can be seen from the context: *...was called a ‘‘stellar and versatile* **acress** *whose combination of sass and glamour has defined her....* The program would need a much better prior model in order to handle this case. When we consider context, we will show how the program can take advantage of the fact that *actress* is more likely than *acres* in the context of *whose*.

Table 4

Example of Rating Candidate Corrections without Context

cor	**%**	**Raw Score**	**freq(cor)**	**Pr(typo│cor)**
actress	37	.157	1343	55./470,000
cress	0	.000	0	46./32,000,000
caress	0	.000	4	.95/580,000
access	0	.000	2280	.98/4,700,000
across	18	.077	8436	93./10,000,000
acres	21	.092	2879	417./13,000,000
acres	23	.098	2879	205./6,000,000

4. Evaluation

Many typos such as *absorbant* have just one candidate correction, but others such as *adusted* have multiple corrections. Table 5 shows examples of typos with less than ten candidate corrections. The candidate corrections are ordered by likelihood.

Table 5

Examples of Typos with Various Numbers of Candidate Corrections

#	**Typo**	**Corrections**
0	admininistration	

1	absorbant	absorbent
2	adusted	adjusted dusted
3	ambitios	ambitious ambitions ambition
4	compatability	compatibility compactability comparability computability
5	afte	after fate aft ate ante
6	dialy	daily diary dials dial dimly dilly
7	poice	police price voice poise pice ponce poire
8	piots	pilots pivots riots plots pits pots pints pious
9	spash	splash smash slash spasm stash swash sash pash spas

Most typos have relatively few candidate corrections. The table below shows the number of typos[3] broken out by the number of corrections in seven month-long samples of the AP newswire. In March, for example, there were 720 typos with 0 corrections, 1120 typos with 1 correction, 269 with 2 corrections, etc. The final column shows that there is a general trend for fewer choices, though the 0-choice case is special. (The system was trained on the AP wire from 2/88 - 2/89; the results below were computed from AP wire during 3/89 - 9/89).

Table 6

Number of Typos by Month and Number of Candidate Corrections

3. For the purposes of this experiment, a typo is a lowercase word rejected by the Unix® spell program.

#	March	April	May	June	July	Aug	Sept	Total
0	720	604	542	606	492	465	508	3937
1	1120	997	1037	1007	958	944	930	6993
2	269	224	209	223	199	224	214	1562
3	109	92	89	101	79	87	82	639
4	58	57	62	45	43	59	43	367
5	54	41	20	26	28	24	28	221
6	35	22	19	19	22	17	23	157
7	20	11	13	7	11	15	17	94
8	19	14	14	5	7	7	16	82
9	15	11	6	11	10	8	16	77
10+	154	97	79	75	53	77	78	613
Total	2573	2170	2090	2125	1902	1927	1955	14,742

We decided to look at the 2-candidate case in more detail in order to test how often the top scoring candidate agreed with a panel of three judges. The judges were given 564 triples and a few concordance lines:

absurb absorb absurd

527595: financial community . *E* *S* ‘‘ It is absurb and probably obscene for any person so engaged to und actuall actual actually

3603767:le in determining whether the defendant actuall will die . *E* *S* In the 1985 decision , the justic

741671: was Pearl Harbor Day . *E* *S* The day actuall is Dec. 7 . *E* *S* Bush said the extradition Wednes

The first word of the triple was a *spell* reject; the other two were the candidates, sorted by alphabetical order. The judges were given a 5-way forced choice. They could circle any one of the three words, if they thought that was what the author had intended. Alternatively, if they thought that the author had intended something else, they could write down ‘‘other’’. Finally, if they weren’t sure, they could write ‘‘?’’. The distribution of responses is shown in the following table.

Table 7

Distribution of Judges’ Responses

	Judge 1	**Judge 2**	**Judge 3**
choice 0 (spell error)	99	124	93
choice 1	188	176	167
choice 2	175	159	151
other	28	26	30
?	74	79	123
total	564	564	564

It should be clear from the table above that *spell* is rejecting too many words. In almost 20% of the triples, *correct* was given a non-problem to correct:

acquirees acquirers acquires

1434654:be acquirers , as they have been , than acquirees . *E* *S* If the industrials had attracted bids th

Since we were mostly concerned with evaluating the scoring function, we didn't want to be distracted with errors in *spell* and other problems that are beyond the scope of this paper. Therefore, we decided to consider only those cases where at least two judges circled one of the two candidates, and they agreed with each other. This left 332 triples.

The following table shows that this version of *correct* agrees with the majority of the judges in 87% of the 332 cases of interest. In order to help calibrate this result, three inferior methods are also evaluated. The *channel-only* method selects the candidate with the larger channel probability $Pr(t|c)$. The *prior-only* method selects the candidate with the larger prior $Pr(c)$. Finally, the *chance* method selects the first candidate. As the following table shows, *correct* is significantly better than the three alternative methods. The table also evaluates the three judges.[4] The table shows that the judges significantly out-perform *correct*, indicating that there is room for improvement. All three judges found the task more difficult and more time consuming than they had expected. Each judge spent about half a day grading the 564 triples.

Table 8

Performance without Context

4. Judges were only scored on triples for which they selected ''1'' or ''2,'' and for which the other two judges agreed on ''1'' or ''2.'' A triple was scored ''correct'' for one judge if that judge agreed with the other two and ''incorrect'' if that judge disagreed with the other two.

Method	**Discrimination**	**%**[6]
correct	286/329	87 ± 1.9
channel-only	263/329	80 ± 2.2
prior-only	247/329	75 ± 2.4
chance	172/329	52 ± 2.8
Judge 1	271/273	99 ± 0.5
Judge 2	271/275	99 ± 0.7
Judge 3	271/281	96 ± 1.1

We were also interested in testing whether the score predicted accuracy. Consider, for example, a group of typos whose average score was 80 percent. Perfect accuracy would be achieved if exactly 80 percent of this group agreed with the majority opinion of the judges. Figure 1 shows a test of this concept, although our data are scanty for this purpose. The horizontal axis shows the score from *correct* averaged over a group of 20 typos. The vertical axis shows the fraction of this group that were right (agreed with the majority opinion of the judges). The diagonal line indicates perfection. The curved lines above and below the perfection line show one standard deviation limits for estimating probabilities from samples of 20.

All three of the methods shown are overconfident when their prediction is between 90 percent and 99 percent. The channel-only method is overconfident over the wider range from about 80 percent

6. The standard deviations are calculated assuming a binomial distribution of d drawings with probability n/d, where n is the numerator and d is the denominator shown in column two. The calculation is $100\sqrt{n(d-n)/d^3}$.

to 99 percent, while the prior-only method is overconfident over most of the range. The observations on *correct* are outside of the one standard deviation limits about as much as would be called for by chance, while each of the other two methods has more points outside than would result just by chance. We conclude that the scores from *correct* predict accuracy fairly well; scores from the other two methods are more problematic.

Poor Estimates of Context Offer Little or No Help

	chance	MLE	ELE
wrong	164.5	15	169
uninformative	0	136	4
right	164.5	178	156

A more sophisticated method of ‘‘fixing’’ the zeros based on the Good-Turing (GT) estimate is discussed in Church and Gale (1990). The following table shows that this method performs significantly better than chance.

Table 10

A Careful Measure of Context Helps

	chance	GT
wrong	164.5	45
uninformative	0	4
right	164.5	280

How do we combine this contextual evidence with the other knowledge sources? Recall that we previously scored candidate corrections with $Pr(c)\ Pr(t|c)$. We now enhance this formula to include contextual factors: $Pr(c)\ Pr(context,\ t|c)$. If we assume that the context is determined by the adjacent words, *l* and *r*, and that these factors are independent with each other and with the typo, then we can use the formula: $Pr(c)\ Pr(t|c)\ Pr(l|c)\ Pr(r|c)$.

The following table shows the results when the three estimates of context are combined with the previous version of *correct*. We find, as we expected, that GT estimates of context provide a significant improvement over ignoring the context.[9] Adding the GT estimate of context improves performance from 87% to 90%. In contrast, poor estimates of context, such as ELE, actually *degrade* performance significantly. It is better to ignore context than to use it badly.

Table 11

Context is Misleading Unless Carefully Measured

	no context	+MLE context	+ELE context	+GT context
wrong	43	11	61	34
uninformative	0	136	0	0
right	286	182	268	295

9. The GT method changes the program's preference in 25 of the 329 cases; 17 of the changes are right and 8 of them are wrong. The probability of 17 or more right out of 25, assuming equal probability of two alternatives, is .04. Thus, we conclude that the improvement is significant.

5. Combining Multiple Knowledge Sources

Four knowledge sources have been discussed: (1) the prior, (2) the channel probability, (3) the left context and (4) the right context. Ideally, we would hope that each of these knowledge sources is contributing new (independent) information. Figure 2 shows that this is so. There is a scatter plot for each pair of the four factors. The plot shows the log[10] of the score assigned by one of the factors to the correct word (as determined by our panel of judges). Note that none of the scatter plots shows a strong correlation.

Figure 1. The fraction actually correct is plotted against the predicted score. The prior-only model is shown with long dashes, the channel-only model with short dashes, and *correct* with a solid line. The straight line is ideal performance, and the curved lines show one standard deviation limits. *Correct* is outside the one standard deviation limits about as much as could be expected by chance, while the two partial models are less accurate.

6. Contextual Constraints

We pointed out before that the human judges were extremely reluctant to cast a vote without more

10. A log transform is use here because the factors are combined by a multiplicative rule.

information, and that they were much more comfortable when they could see a concordance line or two. This suggests that contextual clues might help improve performance. However, it is important to estimate the context carefully; we have found that poor measures of context are worse than none.

In this work, we use a simple n-gram model of context, The calculation is $100\sqrt{n(d-n)/d^3}$. The calculation is $100\sqrt{n(d-n)/d^3}$. simple (compared with much more sophisticated methods used in AI and natural language processing), even these simple n-gram methods illustrate the problem that poor estimates of contextual probabilities are worse than none. The same estimation issue is probably even more critical when more sophisticated AI models of context are used.[7]

The following table shows two inferior methods of estimating The calculation is $100\sqrt{n(d-n)/d^3}$. The calculation is $100\sqrt{n(d-n)/d^3}$. Simple attempts to ‘‘fix’’ these zeros can be disasterous; the ELE, for example, badly overestimates the probability of a bigram that has not been seen in the training set, and consequently, it is wrong more often than it is right.

Table 9

10. The MLE statistics suggest that a trigram model such as (Mays, Damerau and Mercer, 1990) might not work much better than the simple bigram model that we have been using. The trigram model might help for the 15 cases where the bigram context is misleading, but it probably won’t help for the much larger set of 136 cases where the bigram context is uninformative, since the trigrams will also be uninformative in these cases.

	channel	prior	left	right
channel	1.00	-0.05	-0.00	0.07
prior	-0.05	1.00	0.15	0.22
left	-0.00	0.15	1.00	0.11
right	0.07	0.22	0.11	1.00

Figure 2. Each panel shows a scatter plot comparing two of our knowledge sources. The logarithms of the scores assigned by the individual factors to the correct word are plotted. The plots show that any two of these factors are nearly independent.

The following table shows the pair-wise correlations. Note that most of the correlations are quite small, although there is a modest correlation between the prior probability and the left and right contextual probabilities. As mentioned above, this lack of correlation is encouraging, because it suggests that each knowledge source is contributing new information, which will (hopefully) improve performance.

Table 12

Correlations of Logarithms of Scores from Individual Knowledge Sources

# of KBS	model	performance	mean score
1	channel	80	75
1	prior	76	73
1	left	78	77
1	right	77	74
2	channel + prior	87	84
2	channel + left	87	85

2	channel + right	88	85
2	prior + left	83	82
2	prior + right	80	79
2	left + right	86	84
3	channel + prior + left	90	88
3	channel + prior + right	88	87
3	channel + left + right	90	89
3	prior + left + right	86	85
4	channel + prior + left + right	90	89

The following table shows that performance is generally improved with each additional knowledge source. That is, each of the knowledge sources by itself perform better than chance, pairs of knowledge sources perform better still, triples are better yet, and all four perform best of all. The results below show that most of this pattern holds, except that we cannot say that the 4-way combination is significantly better than all 3-way combinations because our experiment is too small. We conclude that it is worth seeking additional factors.

The fourth column shows the mean of the scores for the correct words. Note that scores are highly correlated with empirically observed performance, as we would hope.

Table 13

Performance Increases with Number of Knowledge Sources

del[X, Y] = Deletion of Y after X

Y (Deleted Letter)

X	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	7	58	21	3	5	18	8	61	0	4	43	5	53	0	9	0	98	28	53	62	1	0	0	2	0
b	2	2	1	0	22	0	0	0	183	0	0	26	0	0	2	0	0	6	17	0	6	1	0	0	0	0
c	37	0	70	0	63	0	0	24	320	0	9	17	0	0	33	0	0	46	6	54	17	0	0	0	1	0
d	12	0	7	25	45	0	10	0	62	1	1	8	4	3	3	0	0	11	1	0	3	2	0	0	6	0
e	80	1	50	74	89	3	1	1	6	0	0	32	9	76	19	9	1	237	223	34	8	2	1	7	1	0
f	4	0	0	0	13	46	0	0	79	0	0	12	0	0	4	0	0	11	0	8	1	0	0	0	1	0
g	25	0	0	2	83	1	37	25	39	0	0	3	0	29	4	0	0	52	7	1	22	0	0	0	1	0
h	15	12	1	3	20	0	0	25	24	0	0	7	1	9	22	0	0	15	1	26	0	0	1	0	1	0
i	26	1	60	26	23	1	9	0	1	0	0	38	14	82	41	7	0	16	71	64	1	1	0	0	1	7
j	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0
k	4	0	0	1	15	1	8	1	5	0	1	3	0	17	0	0	0	1	5	0	0	0	1	0	0	0
l	24	0	1	6	48	0	0	0	217	0	0	211	2	0	29	0	0	2	12	7	3	2	0	0	11	0
m	15	10	0	0	33	0	0	1	42	0	0	0	180	7	7	31	0	0	9	0	4	0	0	0	0	0
n	21	0	42	71	68	1	160	0	191	0	0	0	17	144	21	0	0	0	127	87	43	1	1	0	2	0
o	11	4	3	6	8	0	5	0	4	1	0	13	9	70	26	20	0	98	20	13	47	2	5	0	1	0
p	25	0	0	0	22	0	0	12	15	0	0	28	1	0	30	93	0	58	1	18	2	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0
r	63	4	12	19	188	0	11	5	132	0	3	33	7	157	21	2	0	277	103	68	0	10	1	0	27	0
s	16	0	27	0	74	1	0	18	231	0	0	2	1	0	30	30	0	4	265	124	21	0	0	0	1	0
t	24	1	2	0	76	1	7	49	427	0	0	31	3	3	11	1	0	203	5	137	14	0	4	0	2	0

u	26	6	9	10	15	0	1	0	28	0	0	39	2	111	1	0	0	129	31	66	0	0	0	0	1	0
v	9	0	0	0	58	0	0	0	31	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	1	0
w	40	0	0	1	11	1	0	11	15	0	0	1	0	2	2	0	0	2	24	0	0	0	0	0	0	0
x	1	0	17	0	3	0	0	1	0	0	0	0	0	0	0	6	0	0	0	5	0	0	0	0	1	0
y	2	1	34	0	2	0	1	0	1	0	0	1	2	1	1	1	0	0	17	1	0	0	1	0	0	0
z	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
@	20	14	41	31	20	20	7	6	20	3	6	22	16	5	5	17	0	28	26	6	2	1	24	0	0	2

add[X, Y] = Insertion of Y after X

Y (Inserted Letter)

X	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	15	1	14	7	10	0	1	1	33	1	4	31	2	39	12	4	3	28	134	7	28	0	1	1	4	1
b	3	11	0	0	7	0	1	0	50	0	0	15	0	1	1	0	0	5	16	0	0	3	0	0	0	0
c	19	0	54	1	13	0	0	18	50	0	3	1	1	1	7	1	0	7	25	7	8	4	0	1	0	0
d	18	0	3	17	14	2	0	0	9	0	0	6	1	9	13	0	0	6	119	0	0	0	0	0	5	0
e	39	2	8	76	147	2	0	1	4	0	3	4	6	27	5	1	0	83	417	6	4	1	10	2	8	0
f	1	0	0	0	2	27	1	0	12	0	0	10	0	0	0	0	0	5	23	0	1	0	0	0	1	0
g	8	0	0	0	5	1	5	12	8	0	0	2	0	1	1	0	1	5	69	2	3	0	1	0	0	0
h	4	1	0	1	24	0	10	18	17	2	0	1	0	1	4	0	0	16	24	22	1	0	5	0	3	0
i	10	3	13	13	25	0	1	1	69	2	1	17	11	33	27	1	0	9	30	29	11	0	0	1	0	1
j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
k	2	4	0	1	9	0	0	1	1	0	1	1	0	0	2	1	0	0	95	0	1	0	0	0	4	0

l	3	1	0	1	38	0	0	0	79	0	2	128	1	0	7	0	0	0	97	7	3	1	0	0	2	0
m	11	1	1	0	17	0	0	1	6	0	1	0	102	44	7	2	0	0	47	1	2	0	1	0	0	0
n	15	5	7	13	52	4	17	0	34	0	1	1	26	99	12	0	0	2	156	53	1	1	0	0	1	0
o	14	1	1	3	7	2	1	0	28	1	0	6	3	13	64	30	0	16	59	4	19	1	0	0	1	1
p	23	0	1	1	10	0	0	20	3	0	0	2	0	0	26	70	0	29	52	9	1	1	1	0	0	0
q	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
r	15	2	1	0	89	1	1	2	64	0	0	5	9	7	10	0	0	132	273	29	7	0	1	0	10	0
s	13	1	7	20	41	0	1	50	101	0	2	2	10	7	3	1	0	1	205	49	7	0	1	0	7	0
t	39	0	0	3	65	1	10	24	59	1	0	6	3	1	23	1	0	54	264	183	11	0	5	0	6	0
u	15	0	3	0	9	0	0	1	24	1	1	3	3	9	1	3	0	49	19	27	26	0	0	2	3	0
v	0	2	0	0	36	0	0	0	10	0	0	1	0	1	0	1	0	0	0	0	1	5	1	0	0	0
w	0	0	0	1	10	0	0	1	1	0	1	1	0	2	0	0	1	1	8	0	2	0	4	0	0	0
x	0	0	18	0	1	0	0	6	1	0	0	0	1	0	3	0	0	0	2	0	0	0	0	1	0	0
y	5	1	2	0	3	0	0	0	2	0	0	1	1	6	0	0	0	1	33	1	13	0	1	0	2	0
z	2	0	0	0	5	1	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	4
@	46	8	9	8	26	11	14	3	5	1	17	5	6	2	2	10	0	6	23	2	11	1	2	1	1	2

Many other factors can be defined. One might first parse the input (Hindle, 1983; Seneff, 1989) and then look at words in particular systactic constructions. For example, there are interesting constraints between verbs and objects which are not always modeled very well by simple n-gram methods. Of course, we should also consider extending the context to trigrams, which has been useful in speech recognition (Jelinek, 1985), and could be useful here.[11]

11. Mays, Damerau and Mercer (1990) have used a trigram model for spelling correction. It would be interesting to know how much the extra context improves performance.

However, since zero is the most common bigram frequency already, trigrams will not help in about half of the cases. Parts of speech (Church, 1989; Elbeze, 1990) of the adjacent words should also be investigated. Thesaurus distances have proved useful in information retrieval applications (Salton, 1989, 9.6), and could provide additional information here. Morphological decomposition might also be worth investigating.

As more factors are added, we believe the methods for combination need to be studied more closely. Each of our individual factors is overconfident, so it would be reasonable to shrink each likelihood ratio towards 1 before multiplying. The least overconfident sources of evidence would then get greater weight. It would also be reasonable to shrink individually poorly measured factors.

7. Conclusions

There have been a number of spelling correction programs in the past such as Kucera (1988) that generated a list of candidates by looking for insertions, deletions, substitutions and reversals, much as we have been doing here. Our contribution is the emphasis on scoring. McIlroy, the author of the Unix® *spell* program (1982), intentionally focused on the spelling detection problem, and argued (private communication) that spelling correction was a bad idea so long as the corrector couldn't separate the plausible candidates from the implausible ones. He felt that it was probably more distracting than helpful to bury the user under a long list of mostly implausible candidates. In this work, we have attempted to show that it is possible to sort the candidates by a likelihood function that agrees well enough with human judges to be helpful.

We have furthermore studied the problem of adding contextual information. We have looks at just a few of the more obvious approaches. We have considered only the two adjacent words to the typo, and have tried to estimate probabilities of the proposed corrections conditional on these adjacent words. Using the simplest estimators that one might think of, the maximum likelihood and expected likelihood estimates, actually degrade performance. Fortunately, if the context is estimated very carefully with Good-Turing methods, then context does improve performance, as one would hope it should.

We noticed that the human judges were extremely reluctant to cast a vote given only the information available to the program, and that they were much more comfortable when they could see a concordance line or two. Furthermore, the judges all exceeded the program's performance. Apparently the judges are using additional contextual factors. It might be worth looking at a number of possibilities such as parse trees, word trigrams, part of speech n-grams, thesaurus relations, and morphological structures. Many of these possibilities have proved useful in other applications, and may also improve performance in this application, as well, though it may not be easy to make the case given the estimation issues. Researchers in both artificial intelligence and statistics should take it as a challenge to show how their methodologies can be used to help raise performance up to the level of human performance.

Acknowledgements

Mark Kernighan worked on preliminary stages of this research.

References

Box G. E. P. and G. C. Tiao (1973) *Bayesian Inference in Statistical Analysis*, Reading, Massachusetts: Addison-Wesley

Chapman R. (1977) *Roget's International Thesaurus*, fourth edition, New York: Harper & Row

Church K. W. (1989), ‘‘A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text,’’ *Proceedings, IEEE International Conference on Acoustics, Speech and Signal Processing, 1989*, Glasgow

Church K. W. and W. A. Gale (1990) ‘‘Enhanced Good-Turing and Cat-Cal: Two New Methods for Estimating Probabilities of English Bigrams,’’ to appear in *Computer, Speech, and Language*

Elbeze M. and A-M. Deroualt (1990) ‘‘A Morphological Model for Large Vocabulary Speech Recognition,’’ *Proceedings, IEEE International Conference on Acoustics, Speech and Signal Processing, 1990*, Albuquerque

Gale W. A. and K. W. Church (1990) ‘‘What’s Wrong with Adding One?’’ submitted to *IEEE Transactions on Acoustics, Speech, and Signal Processing*

Hanks P., T. Long, L. Urdang, (eds.) (1979) *Collins Dictionary of the English Language*, London: Collins

Hindle D. (1983) ‘‘User manual for Fidditch, a deterministic parser,’’ Naval Research Laboratory Technical Memorandum 7590-142

Good I. J. (1953) ‘‘The population frequencies of species and the estimation of population parameters,’’ *Biometrika*, v. 40, pp. 237-264

Jelinek F. (1985) ‘‘Self-organized Language Modeling for Speech Recognition,’’ IBM Report

Kucera H. (1988) ‘‘Automated Word Substitution Using Numerical Rankings of Structural Disparity Between Misspelled Words & Candidate Substitution Words,’’ U. S. Patent Number: 4,783,758

Mays M., F. Damerau and R. Mercer (1990) ‘‘Context Based Spelling Correction,’’ IBM internal memo, RC 15803 (#730266).

McIlroy M. (1982) ‘‘Development of a Spelling List,’’ *IEEE Transactions on Communications*, Vol. COM-30, No. 1

Nàdas A. (1984) ‘‘Estimation of probabilities in the language model of the IBM speech recognition system,’’ *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. ASSP-32 pp. 859-861

Salton G. (1989) *Automatic Text Processing*, Reading, Massachusetts: Addison-Wesley

Seneff S. (1989) ‘‘Probabilistic Parsing for Spoken Language Applications,’’ presented at the International Workshop on Parsing Technologies, Carnegie Mellon University

Sinclair J., P. Hanks, G. Fox, R. Moon, P. Stock, et al. (eds.) (1987) *Collins Cobuild English Language Dictionary*, London: Collins

8. Appendix: Confusion Matrices

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X	Y (correct)																								
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

rev[X, Y] = Reversal of XY

X	Y																								
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	2	1	1	0	0	0	19	0	1	14	4	25	10	3	0	27	3	5	31	0	0	0	0	0
b	0	0	0	0	2	0	0	0	0	0	0	1	1	0	2	0	0	0	2	0	0	0	0	0	0	0
c	0	0	0	0	1	0	0	1	85	0	0	15	0	0	13	0	0	0	3	0	7	0	0	0	0	0
d	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	1	0	0	2	0	0	0	0	0
e	1	0	4	5	0	0	0	0	60	0	0	21	6	16	11	2	0	29	5	0	85	0	0	0	2	0
f	0	0	0	0	0	0	0	0	12	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
g	4	0	0	0	2	0	0	0	0	0	0	1	0	15	0	0	0	3	0	0	3	0	0	0	0	0
h	12	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
i	15	8	31	3	66	1	3	0	0	0	0	9	0	5	11	0	1	13	42	35	0	6	0	0	0	3
j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
k	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l	11	0	0	12	20	0	1	0	4	0	0	0	0	0	1	3	0	0	1	1	3	9	0	0	7	0
m	9	0	0	0	20	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	4	0	0	0	0	0
n	15	0	6	2	12	0	8	0	1	0	0	0	3	0	0	0	0	0	6	4	0	0	0	0	0	0
o	5	0	2	0	4	0	0	0	5	0	0	1	0	5	0	1	0	11	1	1	0	0	7	1	0	0
p	17	0	0	0	4	0	0	1	0	0	0	0	0	0	1	0	0	5	3	6	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	12	0	0	0	24	0	3	0	14	0	2	2	0	7	30	1	0	0	0	2	10	0	0	0	2	0

s	4	0	0	0	9	0	0	5	15	0	0	5	2	0	1	22	0	0	0	1	3	0	0	0	16	0
t	4	0	3	0	4	0	0	21	49	0	0	4	0	0	3	0	0	5	0	0	11	0	2	0	0	0
u	22	0	5	1	1	0	2	0	2	0	0	2	1	0	20	2	0	11	11	2	0	0	0	0	0	0
v	0	0	0	0	1	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
w	0	0	0	0	0	0	0	4	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	8	0
x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
y	0	1	2	0	0	0	1	0	0	0	0	3	0	0	0	2	0	1	10	0	0	0	0	0	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0