Mitton, Roger. (2009) Ordering the suggestions of a spellchecker without using context. *Natural Language Engineering* **15** (2), 173-192.

Natural Language Engineering **1** (1): 1–**??**.   Printed in the United Kingdom

Ordering the suggestions of a spellchecker without using context

ROGER MITTON
School of Computer Science and Information Systems, Birkbeck, University of London, London WC1E 7HX
email R.Mitton@dcs.bbk.ac.uk

(*Received May 2006; revised May 2007, April 2008*)

Abstract

Having located a misspelling, a spellchecker generally offers some suggestions for the intended word. Even without using context, a spellchecker can draw on various types of information in ordering its suggestions. A series of experiments is described, beginning with a basic corrector that implements a well known algorithm for reversing single simple errors, and making successive enhancements to take account of substring matches, pronunciation, known error patterns, syllable structure and word frequency. The improvement in the ordering produced by each enhancement is measured on a large corpus of misspellings. The final version is tested on other corpora against a widely used commercial spellchecker and a research prototype.

1 Introduction

Having located a misspelling, a spellchecker[1] will generally offer a short list of suggested corrections, often in order with the best guess at the top. In ordinary use, a spellchecker is called upon to check whole passages of text, in which the misspellings will appear in context. However, a spellchecker can get a long way without using context and in fact many spellcheckers confine themselves to isolated-word correction even though the context is there for them to use (Kukich 1992).

Even without context, there are various types of information a spellchecker can use in making its list. At the least, a spellchecker can compare the misspelling with a dictionary of correct spellings, treating the spellings simply as character strings. But it can also use information about typical misspellings or other aspects of words such as pronunciation or word frequency. In this paper I assess the value of using

[1] There is some awkwardness about the term ‘spellchecker’. The structure of the word seems to require a noun as the first part (compare ‘gatekeeper’) but ‘spell’ as a noun refers to a magical incantation or a period of time; in the context of orthography ‘spell’ is only a verb. But ‘spelling checker’ is cumbersome, and putting a space in (‘spell checker’) is no help. The more widely used ‘spellchecker’ has now been accepted by most modern dictionaries as the standard term for this item of software.

these extra resources by developing a corrector in stages, adding modules to take account of various types of information and evaluating the contribution of each.

For the purposes of this paper I will assume that a misspelling has been detected, one way or another, and that we are focussing on the production of a list of suggestions. I do not mean to suggest by this that the detection, as opposed to the correction, of misspellings is an insignificant operation. Far from it. It is a non-trivial problem with which current spellcheckers have only partial success (Pedler 2001), and a good deal of effort has gone into detecting real-word errors — the kind of error that results from writing the wrong word, such as *there* for *their* — for which the use of context is obviously essential (Mays et al. 1991; Golding 1995; Golding and Schabes 1996; Golding and Roth 1999; Carlson et al. 2001; Hirst and Budanitsky 2005). But this paper concentrates on the correction part of the task.

The work described here is specific to English. The ideas could be used for other languages but would have to be adapted each time.

2 A test corpus

Since I am making no use of context, a suitable test corpus needs to contain no more than a list of misspellings, each one associated in some way with its correct spelling. Some years ago I gathered together a number of collections of misspellings which various people had assembled in the course of their research (Mitton 1985). Some were from schoolchildren, some from university students, some from adult literacy students. Some were American, some British. (There were also some from non-native speakers, though I excluded these from the research reported here.) Most were from spelling tests though some were from free writing; the spelling tests produced large numbers of misspellings of relatively few words while the free writing produced misspellings of many words, but only a handful for each one. I had them keyed in and wrote some documentation on each file; they are available from the Oxford Text Archive (website `ota.ahds.ac.uk`).

For the present research, I combined all those from native speakers into a single file, in the same format. I excluded American forms and their misspellings. Since I was interested in the range of errors rather than their frequency, I removed duplicates; *definate,* for example, though a popular misspelling of *definite,* appears only once. (A given misspelling might occur more than once in the corpus but only as a misspelling of different words.) The misspellings are not necessarily non-words; ten percent are dictionary words, such as *warship* for *worship.*

The great majority of the misspellings were originally handwritten. This might appear to limit the usefulness of this corpus for spellchecking research, since the misspellings that spellcheckers have to deal with are, of course, keyboarded. Since my main interest was in helping poor spellers rather than correcting typos, this was not a problem for me. But, in any case, it is not as much of a limitation as might at first appear. As I discuss in the next section, misspellings produced by poor spellers are orthographically more remote from their targets than those caused by typing slips, and consequently present a more severe challenge. A spellchecker that can deal with poor spellings has a good chance of handling typos, but the reverse is

Table 1. *Extract from the corpus*

`$ambulance`
`amberlance`
`ambuemce`
`ambulence`
`$amendment`
`ammendment`
`$amiable`
`abaimable`
`aimabable`
`aimabial`
`aimable`
`aimably`
`aimaible`
`aimeable`
`aimealable`

less likely to be true. Besides, the techniques I describe could easily be adapted to deal more specifically with typing errors, if that was required. I return to this point later.

The corpus I have worked with contains 35,610 misspellings of 5,964 words. This actually excludes a small number (172) of the correct words in the original collection since they were not in my spellchecking dictionary (a dictionary derived from the Oxford Advanced Learner's Dictionary of Current English (Mitton 1986), also available from the Oxford Text Archive). I excluded these and their misspellings from analysis as there was little point in establishing repeatedly that they were not in the dictionary. Most of these missing items, though correct 'words' in the context of the corpus, are not items one would normally expect to find, in that form, in a standard dictionary. They include, for example, proper nouns *(Eastwood, Sheba)*, trade names *(thermawear)*, words with apostrophes *(headmistress's)*, made-up words *(telltaling, unfavourites)*, hyphenated forms *(cabbage-looking, ex-policewoman)*, and two-word items *(too much)* where the misspelling was a single string *(tomuch)*. Table 1 shows a small section of the corpus; each correct word is followed by the misspellings of that word, the correct words being marked with a $. (The corpus contains 72 versions of *amiable*; only the first few are shown here.)

Most of the items in the corpus are misspellings that a spellchecker might reasonably be expected to correct, like those in Table 1. But the corpus also contains the efforts of young children and extremely poor spellers being subjected to spelling tests way beyond their ability, so there are some misspellings — an extreme example is the single letter *o* as a spelling of *accordingly* — that you would not expect any spellchecker to cope with.

This corpus and the others that I describe later are available for download from `www.dcs.bbk.ac.uk/~roger`.

3 A simple corrector

Most research on spellchecking has been done on typewritten input, and it has been known for many years (Damerau 1964; Pollock and Zamora 1984) that the great

Table 2. *All the simple-error variants of 'pord'*

```
aprod .. zpord,
   paord .. pzord,
   poard .. pozrd,
   porad .. porzd,
   porda .. pordz
aord .. zord,
   pard .. pzrd,
   poad .. pozd,
   pora .. porz
ord, prd, pod, por
oprd, prod, podr
```

majority of typing mistakes result in words that contain just a single simple error, a simple error being defined as follows:

1. Omission of one letter — `typwriter`
2. Substitution of one letter — `typeqriter`
3. Insertion of one letter — `typewritwer`
4. Transposition of adjacent letters — `tyepwriter`

A straightforward algorithm for correcting such errors is simply to generate variations on the misspelling, look each of them up in the dictionary and keep any that turn out to be dictionary words (see the work of Gorin, described in Peterson (1980)). For example, given the misspelling *pord,* we first generate all the strings that would give rise to *pord* if we omitted just one letter. We would get *pord* from *apord* by omitting the *a*, and likewise from *bpord* and so on down to *zpord.* Similarly *paord* to *pzord* and so on, ending with *porda* to *pordz.* Then we generate all the strings that would produce *pord* by substitution of one letter (*aord, bord* and so on), then by insertion of one letter (*ord* would produce *pord* if we inserted a *p*) and finally by transposition of adjacent letters. The full list is shown in Table 2.

We keep the ones that match dictionary words — *pored, cord, ford, lord, word, pond, pore, pork, porn, port, pod,* and *prod.* (There are more efficient ways of achieving the same result; see, for example, Oflazer (1996), Savary (2002) or Mihov and Schulz (2004).)

Some spellcheckers assume that the first letter of the misspelling is correct, which it usually is (Yannakoudakis and Fawthrop 1983), to save themselves the bother of looking up words from all parts of the dictionary. The version I used, however, tried all variants of the misspelling, including variations on the initial letter.

Applying this method to the corpus gave a success rate of 33.0%. By 'success' here I mean simply that the correct word appeared somewhere in the list of suggestions. A few more (0.2%) could be caught by the same method if spaces, hyphens and apostrophes were inserted into the variant strings as well as letters; *didn't,* for example, which is in the dictionary in that form, would be retrieved as a suggestion for *didnt.* The proportions of the four simple error types, in those misspellings that contained just one simple error, are presented in Table 3.

This success rate of 33% is substantially lower than the figures of 80% (Damerau 1964) and over 90% (Pollock and Zamora 1984) reported for other corpora. The

Table 3. *Error types in misspellings containing just one simple error*

Substitutions	42%	
Omissions	33%	
Insertions	19%	
Transpositions	6%	
Total (= 100%)	*11769*	*(= 33.0% of all the misspellings)*

reason is probably that those corpora consisted largely of typing errors whereas mine contains mostly spelling errors, many produced by people in spelling tests. The proportions of the four error types also differ between different corpora. Substitutions seem to top the list in spelling errors whereas omissions and insertions are more common in typos. (Substitutions seem to predominate also in errors caused by optical character readers, faulty transmission lines and so on (Damerau 1964; Sun et al. 1992) though presumably the precise pattern of errors will vary with the technology.)

For a high proportion of the misspellings (44%) this method produced no suggestions at all. For 21% it produced just one suggestion, for 8% it produced two and so on steadily downwards until for 0.4% it produced more than forty. This last group consisted largely of single-letter misspellings; every other letter of the alphabet (upper and lower case) was a possible substitution and all letters of the alphabet appear as entries in the dictionary. Lists of over forty were also produced for *con, lan* and *san.*

Obviously there are many misspellings for which this simple algorithm will fail to find the correction. If *pord* appeared in 'She *pord* out the tea,' the right word — *poured* — will not appear in the list. A corrector needs to cast its net more widely.

4 Assembling a collection of candidates

A different way to approach the task of correcting a misspelling is to gather a collection of words from the dictionary that look as though they might be the intended word and then to put them in order, in some way, so that the best guess is at the top, the next best is second and so on. How to assemble this collection? Given a dictionary of, say, 70,000 words on the one hand and a misspelling such as *aimabial* on the other, how is the spellchecker going to assemble a collection of a few hundred promising-looking words, hopefully containing *amiable*?

One way is to use a key somewhat like that developed in the SPEEDCOP project (Pollock and Zamora 1984). The Soundex system (Odell and Russell 1918; Davidson 1962; Knuth 1973) is an earlier version of the same basic idea. We take the misspelling and generate a compressed version of it — a key — that contains what we hope are the salient features — the features that it is most likely to have in common with the target. For example, people often write double consonants as single, or vice-versa; we retain just a single consonant in the key, so that, for instance, *disapoint*, *dissapoint* and *dissappoint* will have the same key as *disappoint.* People often have trouble with vowels (*seperate*, *definate*) so we simply exclude non-initial

Table 4. *Key-generating algorithm*

1.	Keep the first letter, though merge some initial letters together: {*a, e, i*}, {*o, u*}, {*g, j*}, {*q, c*}, {*f, v*}, {*w, r*}, {*z, x, s*}.
2.	Discard subsequent vowels and *h, r, w* and *y*.
3.	Convert double letters to single and *ck* to *c*.
4.	Remove *t* from *tch*, *d* from *dg*, *s* from *xs* and *c* from *xc* before *i, e,* or *y*.
5.	Merge some more groups of letters together: {*m, n*}, {*g, j*}, {*c, k*}.
6.	Finally truncate the resulting string to a maximum of five letters.

Table 5. *Extract from the keyed dictionary file*

tftd	`tufted`	*(non-initial vowels omitted from key)*
tfts	`thefts`	*(non-initial h likewise)*
tfts	`tufts`	
tftst	`thriftiest`	*(example of a one-word group)*
tftl	`thriftily`	*(tftl follows tftst in key ordering)*
tftls	`thriftlessness`	*(key no longer than five letters)*
tftls	`thriftlessly`	
tftls	`thriftless`	
tfc	`trafficker`	*(tfc follows tft in key ordering)*
tfc	`traffic`	
tfc	`terrific`	
tfcd	`trafficked`	*(ck treated as double c)*
tfcd	`two-faced`	

vowels from the key. The key-generating algorithm that I have developed is shown in Table 4. For *aimabial* it produces *ambl*.

Using the same key-generating algorithm, we produce a key for every word in the dictionary and we create a version of the dictionary sorted on these keys, so that all words with the same key are grouped together and groups of words whose keys start with the same letters are nearby. We look up the key of the misspelling in this dictionary of keys. We pull in all the words that have the same key as the misspelling, plus some more on either side. The key for *amiable* is *ambl* so it would appear in the list of candidates for *aimabial*.

The ordering used for the keys is not actually alphabetical order, but the following: *vfpbdtqkcxszgjnmlrwaeihouy*. The idea is to put the keys of misspellings closer to the keys of their targets. For example, people do not often write a *b* instead of a *c* (or vice-versa), but they do sometimes write a *b* instead of a *p*. Suppose someone has written 'The crowd *disbersed*.' The key of the misspelling is *dsbsd*. If the keys of the dictionary words were ordered alphabetically, the key of the target *dispersed*, which is *dspsd*, would be some distance away since *b* and *p* are widely separated in the alphabet. By using this alternative ordering, *dsbsd* and *dspsd* are close together, making it more likely that *dispersed* will appear in the list of candidates for *disbersed*.

For the dictionary I used, the key-generating algorithm generated over 25,000 different keys. The key groups vary a good deal in size. The largest group — those with *cs* as the key — has 183 words; at the other extreme there are over 10,000 one-word groups. Table 5 shows a small part of the keyed dictionary.

This system copes well with a range of omissions and substitutions, but not so well with insertions and transpositions. So, as well as producing a key from the original misspelling, it generates variations on the misspelling and produces a key for each of these. Taking the example of *pord* used earlier, it would generate *ord, prd, pod* and *por* to cope with a possible insertion, and *oprd, prod* and *podr* to cope with a possible transposition, producing a key for each of these variations. (Some of these keys will be the same — *prd, pod, prod* and *podr* for example, all produce *pd* because non-initial vowels and *r*'s are ignored.) It pulls in words for each of these keys and adds them to the collection of candidates.

Testing this system on the corpus of misspellings described earlier, the collections of candidate words averaged about six hundred; the smallest collections were in the low hundreds and the largest just over a thousand.

The system had considerable success in retrieving the required word. For 74.2% of the misspellings, the required word was in the collection of candidates. (Bear in mind that this corpus does not contain duplicates of common misspellings and that many of the misspellings are extremely remote from their target word.)

Not surprisingly, its successes largely included those of the simple method described earlier. Any target word that appeared in the list retrieved by the simple method was likely to appear also in the (much larger) list retrieved by the key system. There were some exceptions, however, and combining the lists produced by the two methods gave a success rate of 76.9%.

5 Ranking the candidates

Getting the target word into the collection of candidates is only the first step. A spellchecker that offered the required word buried somewhere in a list of several hundred suggestions would not be much use. How is the spellchecker to assess how good each of these candidates is?

With few exceptions, all the 51 misspellings of *bicycle* in the corpus are obviously misspellings of *bicycle* rather than of any other word. The following are a sample: *bicecyle, biclyce, bicycel, bikecule, biycicle, bycicyle.* They have the correct first letter; they have most of the other letters, and they have quite a few in the right order.

This suggests a simple algorithm. We try to match up the letters in the candidate one for one with those in the misspelling and we count any that are left unmatched. If we do this with *bicycle* and, say, *bikecule*, we get five because we are left with a *y* and a *c* from *bicycle* unmatched in *bikecule* (*bikecule* has one *c* but *bicycle* has two), and a *k*, an *e* and a *u* in *bikecule* unmatched in *bicycle.* Then we do the same with letter pairs; *bikecule* fares rather badly here — *bicycle* has the pairs *ic, cy, yc* and *cl* all missing from *bikecule*, and *bikecule* has *ik, ke, ec, cu* and *ul* all missing from *bicycle*, giving nine unmatched pairs. If the first letter is different, we add on some more penalty points, say two more; *bikecule* begins with the right letter so its final score is 5 + 9 giving 14. (There are many variations on this idea of matching letters or groups of letters, mostly more complicated than this one — see, for example Kukich (1992) or Angell et al. (1983).)

The higher the score, the worse the match; a perfect match (comparing *bicycle*

Table 6. *Position of target word in list of suggestions, with letter-matching algorithm*

Target word first in list of suggestions	33.2%
In top three	47.4%
In top six	55.1%
In top ten	60.0%
Retrieved from dictionary but listed below 10th	16.9%
Not retrieved from dictionary	23.1%
Total misspellings = 100%	*35610*

with *bicycle*) would score zero. The other attempts at *bicycle* in the list above look as though they ought to get lower scores than *bikecule* and indeed they do — some score four, others six.

If we do this with each of the candidates for a given misspelling, then each will receive a score and we can then put the candidates in order, the ones with the lower scores being the ones that match the misspelling most closely and which are therefore most likely to be the word required. For example, for *bikecule* we get *buckle* at the top of the list with twelve, then a group with fourteen — *Beccles, bickers, bicycle, bucked, bucket* and a few more — then a group with sixteen and so on. (The ordering of candidates that have the same score is not significant.)

Table 6 shows the results of applying this algorithm to the candidate lists for all of the misspellings in the corpus. It shows that, for 33.2% of the misspellings, the algorithm offered the required word as the first choice in its list; for 47.4% the required word came in the top three suggestions, and so on. For 23.1% of the misspellings, the algorithm had no chance of putting the required word at the top of the list since the required word was not in the set of candidates retrieved from the dictionary in the first place.

The performance of this algorithm is strongly related to the length of the misspelling. If we consider the number of times it offers the required word as the first in its list, it manages this only 13% of the time when the misspelling is four letters or less, but this rises steadily to 76% for misspellings of fifteen letters or more.

How much improvement does Table 6 represent over the simple system described earlier? The two systems do not lend themselves to direct comparison. The first system simply produces a list of candidates, generally short but in no particular order; the second retrieves a much larger list but puts it into order. However, if we say that the second system has achieved a success if it offers the target word in its top ten suggestions, then the success rate has risen from 33% to 60%.

6 Minimum edit distance

Consider the misspelling *highdrollick.* This is obviously meant to be *hydraulic.* The algorithm just described, however, does not offer *hydraulic* as its first suggestion, or even in its first twenty suggestions. Its first choice is *highroad*, with a score of sixteen, then *highball, highbrow, highflier, highjack* and *highroads,* all with eighteen. What is it that seems obvious to us but which the algorithm is clearly missing?

High is pronounced like *hy* — *i* and *y* are pronounced the same in certain contexts and *gh* in certain contexts is silent. The *au* in *hydraulic* is often pronounced like the *o* of *doll.* People often write a double letter for a single, like *ll* for *l; ck* for *c* is the same sort of error. Taking all this into account, *hydraulic* is a strong candidate for *highdrollick.* Since no other word matches *highdrollick* nearly so well, we settle on *hydraulic* as the clear favourite.

How can we get a spellchecker to use the same sort of information so that it will, at least, include *hydraulic* in its list of suggestions and, better still, put it at the top? The method of minimum edit distance, generally attributed to Levenshtein (1966), further developed by Wagner and Fischer (1974) and Veronis (1988), and described in detail in my book (Mitton 1996), provides a mechanism for this.

Briefly, we take a misspelling — say *scmlberd* — and a suggested possible correction — say *scrambled* — and we work out how we can get from one to the other with the minimum amount of editing. By 'editing', I mean inserting a letter, deleting a letter, substituting one letter for another, or transposing adjacent letters. We can get from *scrambled* to *scmlberd* by deleting *ra,* transposing the *bl* and inserting an *r.* If we put a notional cost on each of these operations, we arrive at a total cost for editing *scrambled* into *scmlberd.*

We can regard this cost as a sort of distance. The more editing operations we need, to get from one string to another, or the more costly the editing operations are, the more remote from each other the strings are. *Scmlberd* and *scrambled* are medium close. Not as close as, say, *sissors* and *scissors* but closer than *dufleerth* and *beautiful.* (All these examples are taken from the corpus.)

This provides a method by which a spellchecker can take account of the sort of things I described in my analysis of *hydraulic.* The spellchecker can be primed with the information that *i* for *y* is quite a likely substitution — either in general or for this word in particular — and will therefore attach only a low cost to this. Likewise it can be told to attach a low cost to the insertion of *gh* in words that have a fragment that rhymes with *high,* a low cost also to the substitution of *o* for the *au*, to the insertion of an extra *l* or of a *k* after the *c.* The result is that, when it computes the distance between *highdrollick* and *hydraulic*, it comes out with a low figure — it considers them to be quite close.

In the *highdrollick* example, the pronunciation of the misspelling would match that of the target word, but this is not always the case (Mitton 1987). *Rember* and *remeber* are common misspellings of *remember,* for example, as is *lastest* for *latest.* See also *scmlberd* and some of the attempts at *amiable* in Table 1. The method can cope just as well with errors unrelated to pronunciation, so long as they can be anticipated — a spellchecker can be primed to attach a low cost to the omission of *em* in *remember* or to the insertion of an extra *s* in *latest.* The reason that people have trouble with spelling in English is that English spelling is quirky. Priming the dictionary in this way provides the corrector with information about the quirks.

Of course the minimum-edit algorithm can only home in on the target word if the target word is in the collection retrieved from the dictionary. So if the dictionary is primed with information about the places where people are likely to omit letters or to substitute one letter for another, it is necessary to take account of this

Table 7. *A second extract from the keyed dictionary file*

ftn	`fortune`	
ftn	`fourteen`	
ftn	`furthermore`	*(m and n merged)*
ftn	`phaeton`	*(ph might be misspelt as f)*
ftn	`photon`	
ftnbl	`fissionable`	*(ss might be misspelt as t, as in nation)*
ftnd	`fathomed`	
ftnd	`fattened`	
ftnd	`frightened`	*(g likely to be omitted)*

Table 8. *Position of target word in list of suggestions, with minimum-edit algorithm*

Target word first in list of suggestions	47.2%
In top three	61.5%
In top six	68.3%
In top ten	72.2%
Retrieved from dictionary but listed below 10th	12.2%
Not retrieved from dictionary	15.3%
Total misspellings = 100%	*35610*

information when building the keyed version. For example, the entry for *knuckle* contains the information that the *k* is likely to be omitted. Two keys are generated for *knuckle* — one with the *k* and one without: *(kncl* and *ncl)* — so *knuckle* appears in two places in the keyed version of the dictionary. Now, if the misspelling is *nuckle* (or *nucle* or *nukel* or any variation with the key *ncl*), *knuckle* will appear in the collection of candidates.

Given these extra entries, there is a better chance of getting the required word in the collection of entries retrieved from the dictionary. In fact its success rate on the corpus is 84.7%, a useful improvement on the 76.9% that was reached without the extra entries.

Table 7 presents a second extract from the keyed dictionary, this time illustrating some of these extra entries (*phaeton* for example will also appear under *ptn*).

Given a dictionary primed with this sort of information, the task of the minimum-edit scoring system is to bring the target word as close as possible to the top of the list. Table 8 shows how much success it had.

While the simple-error method achieved 33% success and the letter-matching algorithm 60%, the minimum-edit system put the required word in its top ten suggestions for 72% of the misspellings. More than half of its failures were due to the candidate-collection process — the required word was missing from the list of words it was given in the first place. Though occasionally it placed the target word way down the list, the system generally did well. When it was provided with the required word somewhere in its initial list of (several hundred) candidates, the minimum-edit system placed it among the top ten suggestions for 85% of the misspellings, and more often than not at the top of the list.

7 Anticipating errors

How did we get all this information into the dictionary — that *hy* might be misspelt as *high* and so on? In my earlier work (Mitton 1996) I observed that people often rely on the pronunciation of a word to help them to spell it, and I described a program to generate a 'phonetic' spelling for each word (my dictionary contains pronunciations) — for example, *yot* for *yacht* — and to calculate the edit distance between the two. If this was greater than zero, the program tried flagging the letters of the correct spelling in various ways, recalculating the edit distance each time, and it retained those flags that provided the minimum edit distance. So *yacht* ended up with markers on the *a, c* and *h* indicating that an *o* might be substituted for the *a* and that the *ch* might be omitted. I then ran the corrector over large corpora of errors, collecting those errors, or patterns of error, on which it performed poorly, and adding more flags.

More efficient ways of achieving a somewhat similar result have been developed since I did this part of the work. Ristad and Yianilos (1998) describe how the costs can be derived by computer from a large corpus, with no need of human intervention. They applied their system to speech recognition. Brill and Moore (2000) describe a different version of this idea for spellchecking. Taking a large corpus of misspellings paired with correct words, their program matches fragments in each pair and counts the frequency with which one is produced for the other, for instance *ent* for *ant*. For example, if the corpus contained *dependent* as a misspelling of *dependant*, this would add one to the *ent/ant* count. Given a misspelling to correct, the program applies a version of the minimum-edit algorithm described earlier, except that it substitutes fragments rather than single letters and decides on the best match on the basis of the frequencies rather than some a priori costs, on the grounds that, the more frequently a substitution appeared in the training corpus, the more likely it is to figure in this misspelling.

This makes no explicit reference to pronunciation, though it makes use of pronunciation indirectly — the reason that *ent* emerges from the training corpus as a frequent misspelling of *ant* is that it often corresponds to an unstressed syllable, as in *dependant*, where the pronunciation gives no help to the speller. (You wouldn't expect the same problem with words like *chant* or *eggplant*.) Toutanova and Moore (2002) have built on this work by taking explicit account of pronunciation. They also count matches of fragments in pairs of misspellings and correct words, but they are matching pronunciations, of the misspelling on the one hand, as estimated by a text-to-speech converter, and of the correct word on the other, taken from a pronunciation dictionary. Given a misspelling to correct, their program produces a pronunciation for it and then matches candidates by the same method as Brill and Moore, but this time with fragments of pronunciations rather than of spellings. An example of their success is offering *gristle* as the first suggestion for the misspelling *grissel* (the Brill and Moore system offers *grizzel*).

Compared to my dictionary-flagging scheme, these more automated systems are a lot more efficient at extracting useful patterns from error corpora. Whether they also outperform it in suggesting corrections is hard to say; the error corpus that

Brill and Moore used both for training and for testing is not public, so it is not possible to run my system over their data. The alternative — developing a version of their system and running it over my data — is less satisfactory, but the appendix describes an attempt to do this.

8 Combining minimum-edit with letter-match

While minimum-edit works better than the simpler letter-matching for most of the misspellings, there are some misspellings that fare better with the letter-matching. Take *biclyce* for example. This is fairly obviously a misspelling of *bicycle* but the minimum-edit method offers *bicycle* at a disappointing eighth place in the list, behind *bucolic, belike, bucolics, bellyache, bakelite, basic* and *Bexley.* (Actually *bakelite* to *bicycle* all get the same score and it is just bad luck that *bicycle* comes last.) The letter-matching method, by contrast, puts *bicycle* at the top of its list for *biclyce* because they have the same letters and with the same one at the front.

This suggests that a combination of the two methods might work even better. There are three elements to the letter-matching — the initial letter, the number of matching single letters and the number of matching letter-pairs. A few experiments made it clear that there was nothing to be gained by taking account of the letter-pairs — the minimum-edit had already taken full account of this — but that small improvements could be made by attaching more weight to the initial letter and the number of single-letter matches.

Augmenting minimum-edit in these ways produced a difference of only about one percent, but it was in the right direction. When compared with minimum-edit alone, some misspellings were handled better, some worse, but more of the first than the second. (The combined method put *bicycle* second in the list for *biclyce*, still behind *bucolic*, but now ahead of *bellyache* and the rest.) The proportion of misspellings for which the target was offered in the top ten — 60% with letter-matching alone, 72% with minimum-edit alone — was now 73%.

9 Homophones

Ten percent of the misspellings were themselves dictionary words, such as *angel* for *angle.* Some of these were caused by people selecting the wrong spelling from a set of homophones, such as *bawd* or *board* for *bored.* Words like *bawd, board* and *bored* were flagged as homophones of each other in the dictionary, so it was possible to detect when a candidate was a homophone of the misspelling and to promote it up the list.

On the whole this had the desired effect; for 180 of the misspellings, the required word moved up the list. On the other hand, for 140, the required word moved down — the promotion of *bowl* for *boll*, for example, caused the required word *ball* to move down. But the upward movement of those that went up was far greater than the downward movement of those that went down; *colonel*, for example, moved up the list for the misspelling *kernel* from eighth place to second, *bored* moved up the list for *bawd* from 27th to fourth, and *choose* moved up the list for *chews* from 29th place to third.

Since there were so few of these, however, relative to the size of the whole corpus, the overall improvement was tiny.

10 Syllables

The number of syllables in a misspelling is a further possible clue to the word required but is not taken explicitly into account by the letter-match or minimum-edit algorithms. The number of syllables in a word is a function of its pronunciation, not its spelling, but in the case of misspellings, except for those that happen to be in the dictionary, we do not know the pronunciation and have to make an estimate of the number of syllables from the spelling.

In my book (Mitton 1996) I present an algorithm for doing this. First we strip apparent prefixes and suffixes from the spelling, perhaps adding to the syllable count as we do so — for example, the suffix *ed* counts as a syllable in *hunted* but not in *chased*. Then we traverse what remains counting groups of vowel-letters, though trying not to count silent *e*'s (*placemat* has two syllables, not three).

The dictionary I was using contained the number of syllables for each word, so, though the syllable-counter was intended for use with misspellings, it was possible to test it on dictionary words. It calculated the right number of syllables from the spelling for 96% of the words. Improving it further would have meant incorporating a table of exception words and there was no point in doing this since misspellings would never be in the table.

It was now possible to promote those candidates that had the same number of syllables as the misspelling. Disappointingly, this made hardly any difference, and the little difference that it did make was in the wrong direction.

However, though misspellings often had fewer syllables than the target word, they rarely had more. I therefore demoted those candidates that had fewer syllables than the misspelling. The effect of this was at least in the right direction, but still very small, improving the success rate by less than one per cent.

11 Word frequency

Consider the misspelling *arigments*. The spellchecker comes up with *arguments* as its first choice, but then has *arraignments, regiments* and *arrangements*, all with the same score. Other things being equal, presumably *arrangements* is preferable to the other two, just because it is a much more common word. And indeed, in this case, this would have been the right decision — *arrangements* was in fact the required word.

At ninety million words, the written part of the British National Corpus provides word frequency data that is adequate for this task. The BNC actually provides a frequency for a word with a part-of-speech tag — one frequency for *surf* (noun) and another for *surf* (verb) for example. For the present purpose I simply summed these, except for the frequencies of ordinary words used as proper nouns. (For

Table 9. *Position of target word in list of suggestions, after adjusting for word-frequency*

Target word first in list of suggestions	56.4%
In top three	69.2%
In top six	74.6%
In top ten	77.6%
Retrieved from dictionary but listed below 10th	7.1%
Not retrieved from dictionary	15.3%
Total misspellings = 100%	*35610*

example I retained the frequency of *thatcher* as an ordinary noun, not its much larger frequency as a proper noun.)

At the least I could reorder those runs of candidates that had the same minimum-edit score, but I found that it further improved the rankings if I also used the frequencies to adjust the scores. After some experimentation I settled on a simple algorithm, dividing words into the following groups by their frequency (i.e. the number of times they occurred in the BNC) — 0–50, 51–200, 201–500, 501–1000, 1001–2000, over 2000. Candidates in the 0–50 group had five added to their cost, 51–200 four and so on. Table 9 shows the effect.

Factoring in word frequency has an appreciable effect. The previous best score of around 74% of misspellings for which the correct word appeared in the top ten suggestions has now gone up to over 77%. The great majority of the failures are due to the initial stage of trawling the dictionary for possible candidates. If the correct word gets into this initial list, the system makes a good job of putting it near the top, getting it into the top ten over ninety percent of the time and getting two out of three at the very top.

It might be argued that, since many of the misspellings came from spelling tests, this result might simply reflect the fact that the creators of the tests avoided uncommon words when choosing their test words. But the same effect shows up when run on misspellings derived solely from free writing.

12 Different data, and comparisons of performance

Thus far I have been testing the spellchecker on the same data that I have been using to develop it. In general this is not good practice as it runs the risk of tuning the software to the particular features of the test data, producing a spuriously high level of performance which does not generalise to other data sets. So I produced another set of test data, containing broadly the sort of misspellings that my spellchecker was meant to deal with, but from a different source from any of those in the original set.

The book *English for the Rejected* (Holbrook 1964) contains extracts from the creative writing of school pupils in their final years of secondary school. The extracts preserve the spelling and punctuation of the originals. These were pupils who had a poor command of written English. The following samples are representative:

When they are young you have to wate 3 days then you can injeck them for diseases you

Table 10. *Position of target word in list of suggestions, using Holbrook data, with Word for comparison*

	Experimental spellchecker	*Word*
First	57.0%	44.0%
Top three	74.2%	61.6%
Top six	80.9%	67.9%
Top ten	83.9%	70.4%
Total = 100%	*1187*	*1187*

have mack shore they have drye straw when you clean them out you should not leave a falk in with them because the mother might nock it down and the little pigs might stab them souve

The leavs are small The apple fall in Aount time When yow get to the corres inside the corres are pips yow thorght the corres on the fire to brun then.

I created a file containing all 1788 misspelt words from these extracts, with the correct word in each case (except for twenty where I couldn't work out what the correct word was supposed to be). As with my earlier experiments, I recorded how well the spellchecker coped with each misspelling.

For a point of comparison, I also ran them through the spellchecker that comes with Microsoft Word (Office Word 2003, U.K. English) and similarly recorded where the correct words came in its lists of suggestions. (Where the misspelling contained a space, as in *to gether*, I replaced the space with an underscore so that Word tried to correct the whole thing; otherwise it would attempt to correct only the *gether*.) Occasionally, Word offered no suggestions at all, e.g. for *elecenician* (electrician), but it generally offered between one and ten, never more than ten. There were 581 that Word did not consider to be errors; these had to be discounted since Word did not attempt to offer suggestions for them. The results are in Table 10.

The results of this test are reassuring. The spellchecker worked just as well on this data as on the data I used for development, and, compared with a commercial spellchecker in widespread use, it makes better suggestions for a range of misspellings that a spellchecker ought to be able to correct, such as *alloud, eney, brecfast, nabour, secutery, thort, tung* and *whated.*

I chose Word for this test because it is widely used and readily to hand, but a more appropriate point of comparison might be a spellchecker developed by other researchers. In addition, most of the misspellings used for the development of my spellchecker, and all of those in the Holbrook data, were originally handwritten, whereas the misspellings that spellcheckers actually have to deal with are keyboarded. I was able to address both of these points by comparing my results with recent work by Deorowicz and Ciura (2005). They describe an experimental spellchecker and test it on two files of errors, which they have made publicly available. They report impressive results in comparison with several public-domain spellcheckers.

The spellchecker developed by Deorowicz and Ciura has a table of possible substring substitutions and it applies as many as it can to the misspelling, retaining

Table 11. *Position of target word in list of suggestions, using aspell and wikipedia data, with Deorowicz and Ciura for comparison*

	aspell Experimental spellchecker	*aspell Deorowicz and Ciura*	*wikipedia Experimental spellchecker*	*wikipedia Deorowicz and Ciura*
First	71.1%	66.3%	92.9%	94.1%
Top two	83.2%	75.5%	97.2%	97.4%
Top three	88.6%	79.6%	97.9%	98.3%
Top five	91.4%	83.6%	98.6%	98.9%
Top ten	94.4%	85.5%	99.0%	99.0%
Total = 100%	*499*	*511*	*2154*	*2196*

those of the resulting forms that match dictionary words. (In this respect it resembles the work of Brill and Moore (2000) described earlier, but they build their table a priori rather than deriving it automatically from a training corpus.) Suppose, for example, that it had the following substitutions in its table: *ll* to *l*, *ow* to *ou* and *d$* (terminal *d*) to *ed*, and suppose that it was faced with the misspelling *allowd*. By applying all these substitutions, it would find that *allowed* and *aloud* were dictionary words (as well as finding that *alloud, alloued, alowd* and so on were not). They have costs (also assigned a priori) for each of these substitutions, which enables them to place the resulting list of candidates in order of preference. Their method of dictionary storage (a minimal acyclic deterministic finite automaton) enables them both to apply the substitutions and to check the results very rapidly.

The two files of errors that they use as test data are simply lists of misspelt words with their correct forms, without context. Neither contains duplicates. The first — aspell — is derived from a collection of 'hard-to-correct' errors used for testing the GNU Aspell spellchecker (Atkinson 2006). There are 525 misspelt words, each with a single correct form. They seem to be a mixture of typos (*olf* for *old*) and poor spellings (*funetik* for *phonetic*). While some of them are indeed hard to correct (*accosinly* for *occasionally*), many are not (*occurence* for *occurrence*).

The other — wikipedia — is derived from a set of 'typical' spelling errors made by Wikipedia editors (2005). This is larger, with 2240 error words, mostly typos or minor errors, though some more serious (*charistics* for *characteristics*) and a handful that are errors of language or usage rather than spelling (*teached* for *taught*, *Mohammedans* for *Muslims*). For a small proportion of the error words (eight per cent), more than one correct form is provided, for example *villain, villein* and *villi* as correct forms for *villin.*

I rewrote the files in the same format as my own corpus (Table 1). Since my spellchecker uses a British English dictionary, I added British spellings as correct forms. Following Deorowicz and Ciura, errors whose correct form was not in the dictionary were excluded from the analysis, as were errors that matched dictionary words. Where more than one correct form was given for an error, only the highest-ranking candidate was counted in the results (again following Deorowicz and Ciura). For example, *villain* came first in the list of suggestions for *villin* while *villein* came third, so *villin* was counted as an error for which the spellchecker proposed the correct word first in its list.

Table 12. *The percentage of misspellings for which the spellchecker was successful, for successive enhancements of the spellchecker*

	First	*Anywhere/ Top ten**
Reversal of single simple errors	—	33.0%
Search key with letter-match algorithm	33.2%	60.0%
Search key, minimum-edit and letter-match	48.4%	73.4%
With homophones and syllables	49.4%	73.9%
With word frequency	54.2%	76.4%
Total misspellings = 100%	*35610*	*35610*

**For the Reversal method, a success was when the target was included anywhere in its list, regardless of the size of the list. For the other methods, the figures show the proportion of misspellings for which the target appeared as the first suggestion, or in the first ten suggestions.*

Table 11 presents the results of this exercise. The differences in the base totals reflect the different sizes of the dictionaries used by the two spellcheckers. Mine uses a smaller dictionary, so more of the errors were excluded because their target word was not in the dictionary. (This does not mean that they were harder to correct; in fact the errors in them were mostly fairly minor.)

These results are reassuring. Though my spellchecker was directed at correcting poor spellings rather than typing slips, it seems to cope acceptably with keyboarded input. It also bears comparison with a recent research prototype.

13 Conclusion

The results presented in the previous section give some idea of the spellchecker's level of performance, but it was not the primary purpose of this paper to compare this spellchecker with others. The main aim was to assess the amount of improvement in performance to be gained from each of various enhancements. Table 12 brings together some key figures from earlier tables, to show these effects on performance.

Compared to typos, misspellings pose a sterner test for a spellchecker. The simple error-reversal algorithm, which performs well with typos, succeeded with only a third of the misspellings in this corpus. Computing a key from the misspelling and using this to retrieve candidates from a keyed version of the dictionary had much more success in finding the required word, and a simple letter-matching algorithm was often able to place the required word at or near the top of the list. A significant improvement on this was made by priming the dictionary with information about the quirks of English orthography and hence the mistakes that people are likely to make, and using the minimum-edit algorithm to take account of this. Where the misspelling was itself a dictionary word, promoting homophones up the list made a useful contribution, but this was applicable in only a few cases. Attempting to take account of syllable structure made very little difference. Finally, factoring word frequency into the ordering of the suggestions made a further worthwhile improvement.

How much more mileage remains to be made with isolated-word correction? In terms of the challenge that they present to a spellchecker, misspellings range from trivial to impossible. The further you go, the harder it gets.

In the most successful version of the spellchecker, most of the failures occur at the candidate-retrieval stage — the required word is not among the initial set retrieved from the dictionary. This might suggest that more effort should be applied to this process, but I would not be optimistic. With few exceptions, any extra required words retrieved by a wider or cleverer trawl are going to have little in common with their misspellings — that's why they are on the fringe of the candidate collection in the first place — so they are unlikely to make it to the top of the list. If we can't get the required word close to the top, we might as well leave it in the dictionary. But there might be more scope for improving the ordering of the top twenty. This is where I would expect context to help.

14 Appendix: a further comparison

In Section 7, I sketched the system described by Brill and Moore (2000). In order to compare my system with theirs, I implemented a version of the system described in their paper. I used as the training set the main error corpus that I have been referring to in most of this paper and created from it a table of word-fragment substitutions, up to length four, with probabilities (actually three tables, one for word-initial fragments, one for word-ending fragments and one for the rest). For the testing phase, I wrote a module to rank a series of candidate corrections for a given misspelling in order of probability, using these tables. My version of their system should be comparable with the one that produced the row marked 'Max window 3' in Table 2 of their paper.

To make a comparison, I used my own system to generate a list of candidates for a misspelling and then put these candidates into order using my system on the one hand and my version of the Brill and Moore system on the other.

The full Brill and Moore system actually has a further phase in which they employ a 'language model' — they rescore the top five candidates by using a table of word-trigram probabilities. I could not reproduce this since I did not have the data it was based on. In a sense, my system also employs a language model, albeit a simple one, in the use of word frequency, so I disabled the reordering by word frequency from my system for the purpose of these comparisons.

The test data sets were the aspell and wikipedia corpora described above, minus a few items that began with upper-case letters, since my version of the Brill and Moore system did not handle these correctly.

The results are presented in Table 13.

I would not attach much significance to the detail of these results; my implementation of Brill and Moore's system may well differ in many small but perhaps important ways from the original. But they suggest, like the Deorowicz and Ciura

Table 13. *Position of target word in list of suggestions, using aspell and wikipedia data, with Brill and Moore for comparison*

	aspell Experimental spellchecker	*aspell Brill and Moore*	*wikipedia Experimental spellchecker*	*wikipedia Brill and Moore*
First	73.5%	68.3%	93.5%	87.5%
Top three	90.8%	85.2%	98.5%	95.2%
Top six	95.6%	91.9%	99.4%	96.8%
Top ten	97.1%	96.0%	99.7%	97.7%
Total = 100%	*480*	*480*	*2086*	*2086*

results, that my spellchecker at least bears comparison with some other recent experimental systems.

15 Acknowledgements

I am grateful to the anonymous referees for their critical but helpful comments on the first and second submissions and to Sebastian Deorowicz and Marcin Ciura for making available their test data and for providing clarification on some aspects of their experiments.

References

Richard C. Angell, George E Freund, and Peter Willett. Automatic spelling correction using a trigram similarity measure. *Information Processing and Management*, 19(4):255–261, 1983.

Kevin Atkinson. GNU Aspell. `http://aspell.net/`, 2006.

Eric Brill and Robert C. Moore. An improved error model for noisy channel spelling correction. In *Proceedings 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293, 2000.

Andrew J. Carlson, Jeffrey Rosen, and Dan Roth. Scaling up context-sensitive text correction. In *Proceedings 13th Innovative Applications of Artificial Intelligence Conference*, pages 45–50, 2001.

Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the A.C.M.*, 7:171–176, Mar 1964.

Leon Davidson. Retrieval of misspelled names in an airlines passenger record system. *Communications of the A.C.M.*, 5:169–171, Mar 1962.

Sebastian Deorowicz and Marcin G. Ciura. Correcting spelling errors by modelling their causes. *International Journal of Applied Mathematics and Computer Science*, 15(2):275–285, 2005.

Andrew R. Golding. A bayesian hybrid method for context-sensitive spelling correction. In *Proceedings Third Workshop on Very Large Corpora*, pages 39–53, 1995.

Andrew R. Golding and Dan Roth. A Winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34:107–130, 1999.

Andrew R. Golding and Yves Schabes. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *Proceedings 34th Annual Meeting of the Association for Computational Linguistics*, pages 71–78, 1996.

Graeme Hirst and Alexander Budanitsky. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11(1):87–111, 2005.

David Holbrook. *English for the Rejected.* Cambridge University Press, 1964.

Donald E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*, pages 391–392. Addison-Wesley, 1973.

Karen Kukich. Techniques for automatically correcting words in text. *Computing Surveys*, 24(4):377–439, 1992.

V.I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics - Doklady*, 10(8):707–710, Feb 1966.

Eric Mays, Fred J Damerau, and Robert L Mercer. Context based spelling correction. *Information Processing and Management*, 27(5):517–522, 1991.

Stoyan Mihov and Klaus U. Schulz. Fast approximate search in large dictionaries. *Computational Linguistics*, 30(4):451–477, 2004.

Roger Mitton. A collection of computer-readable corpora of English spelling errors. *Cognitive Neuropsychology*, 2(3):275–279, Aug 1985.

Roger Mitton. A partial dictionary of English in computer-usable form. *Literary and Linguistic Computing*, 1(4):214–215, 1986.

Roger Mitton. Spelling checkers, spelling correctors and the misspellings of poor spellers. *Information Processing and Management*, 23(5):495–505, 1987.

Roger Mitton. *English Spelling and the Computer.* Longman, 1996. Available on open access at `eprints.bbk.ac.uk`.

Margaret K. Odell and Robert C. Russell. *U.S. Patents 1261167 (1918), 1435663 (1922).* U.S. Patent Office, 1918.

Kemal Oflazer. Error tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–89, 1996.

Jennifer Pedler. Computer spellcheckers and dyslexics - a performance survey. *British Journal of Educational Technology*, 32(1):23–37, Jan 2001.

James L. Peterson. Computer programs for detecting and correcting spelling errors. *Communications of the A.C.M.*, 23(12):676–687, Dec 1980.

Joseph J. Pollock and Antonio Zamora. Automatic spelling correction in scientific and scholarly text. *Communications of the A.C.M.*, 27(4):358–368, Apr 1984.

Eric Sven Ristad and Peter N. Yianilos. Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998.

Agata Savary. Typographical nearest-neighbor search in a finite-state lexicon and its application to spelling correction. In Bruce W. Watson and Derick Wood, editors, *Proceedings 6th International Conference on the Implementation and Application of Automata*, Springer Lecture Notes in Computer Science 2494, pages 251–260, 2002.

W. Sun, L-M Liu, W. Zhang, and J.C. Comfort. Intelligent OCR processing. *Journal of the American Society for Information Science*, 43(6):422–431, 1992.

Kristina Toutanova and Robert C. Moore. Pronunciation modelling for improved spelling correction. In *Proceedings 40th Annual Meeting of the Association for Computational Linguistics*, pages 144–151, 2002.

Jean Veronis. Computerized correction of phonographic errors. *Computers and the Humanities*, 22:43–56, 1988.

Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the A.C.M.*, 21(1):168–173, Jan 1974.

Wikipedia. `en.wikipedia.org/wiki/Wikipedia:List_of_common_misspellings`, 2005.

E.J. Yannakoudakis and D. Fawthrop. The rules of spelling errors. *Information Processing and Management*, 19(2):87–99, 1983.