Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information Fusion 7 (2006) 207–220

INFORMATION FUSION

www.elsevier.com/locate/inffus

Information fusion approaches to the automatic pronunciation of print by analogy

R.I. Damper [a,b,*], Y. Marchand [b,c]

[a] *Image, Speech and Intelligent Systems (ISIS) Research Group, School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK*

[b] *Institute for Biodiagnostics (Atlantic), National Research Council Canada, Neuroimaging Research Laboratory, 1796 Summer Street, Suite 3900, Halifax, Nova Scotia, Canada B3H 3A7*

[c] *Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada B3H 1W5*

Received 29 March 2004; received in revised form 5 August 2004; accepted 5 August 2004
Available online 11 September 2004

Abstract

Automatic pronunciation of words from their spelling alone is a hard computational problem, especially for languages like English and French where there is only a partially consistent mapping from letters to sound. Currently, the best known approach uses an inferential process of analogy with other words listed in a dictionary of spellings and corresponding pronunciations. However, the process produces multiple candidate pronunciations and little or no theory exists to guide the choice among them. Rather than committing to one specific heuristic scoring method, it may be preferable to use multiple strategies (i.e., soft experts) and then employ information fusion techniques to combine them to give a final result. In this paper, we compare four different fusion schemes, using three different dictionaries (with different codings for specifying the pronunciations) as the knowledge base for analogical reasoning. The four schemes are: fusion of raw scores; rank fusion using Borda counting; rank fusion using non-uniform values; and rank fusion using non-uniform values weighted by a measure of prior performance of the experts. All possible combinations of five different expert strategies are studied. Although all four fusion schemes outperformed the single best strategy, results show clear superiority of rank fusion over the other methods.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Score fusion; Rank fusion; Automatic pronunciation; Analogical reasoning; Speech synthesis

1. Introduction

Text-to-speech (TTS) synthesis is an emerging technology with many potential applications to next-generation computer and information systems [1,2]. A very important sub-problem within TTS synthesis is the automatic generation of word pronunciations from textual input, or ‘print’. Unless we are able to derive a good specification of pronunciation of the individual words in the input, we cannot hope to produce a satisfactory TTS system. Yet for many languages, such as French and English, the relation between letters and sounds can be rather complex and indirect. Since printed text is not always a very direct specification of pronunciation, it is sensible to convert it to something much closer to a representation of the corresponding sound sequence. Linguists have long used the *phoneme* as an abstract unit

[*] Corresponding author. Tel.: +44 023 80 594577; fax: +44 023 80 594498.
E-mail addresses: rid@ecs.soton.ac.uk (R.I. Damper), yannick.marchand@nrc-cnrc.gc.ca (Y. Marchand).

of the sound system of language [1] and pronunciation in TTS systems is most usually represented as a phoneme string. Hence, the process is one of letter-to-phoneme conversion, or translation. Further, although every speaker will say words somewhat differently, and even the same speaker will say the same word differently at different times, it suffices to work with an idealized representation of word pronunciations broadly corresponding to the entries of a pronouncing dictionary; this is called the *phonemic baseform* by Lucassen and Mercer [3]. It is assumed at this stage that a subsequent processing step will adjust the word pronunciations according to sentence context.

Modern TTS systems use look-up in a large dictionary as the primary strategy to determine the pronunciation of input words. However, because language processes are infinitely generative, it is not possible to list exhaustively all the words of a language, so a secondary or 'back-up' strategy is required for the automatic phonemization of words not in the system dictionary. This is a hard computational problem for languages like English and French where there is at best only a partially consistent mapping from letters to sound. Although there exist a wide variety of techniques for pronunciation generation from print [4], currently the best known approach to the problem uses an inferential process of analogy with those words that are listed in the dictionary—so-called pronunciation by analogy (PbA).

One difficulty faced with PbA is that the analogical process yields multiple candidate pronunciations, and some high-accuracy method must be employed to make the correct choice between them. Unfortunately, little or no theory exists to guide this choice. Rather than committing to any one specific heuristic scoring method, it may be preferable to use multiple methods and then employ information fusion techniques to combine them to give a final result, with each heuristic scoring method acting as a "soft expert". Our purpose in this paper is to investigate different approaches to the fusion problem using three dictionaries differing in size and the phoneme coding employed. We concentrate on PbA for English, since this is the language for which TTS synthesis is best developed and it is also one of the hardest languages for which to derive pronunciations automatically.

Our formulation of the fusion problem in this paper is slightly unusual in that it is neither multi-sensor nor is it multi-source, in the sense that there is only a single source of text. However, if we shift emphasis to seeing the 'sources' as the separate analyses performed by the soft experts, then the fusion of these analyses is clearly a problem within the domain of information fusion.

The remainder of this paper is organised as follows. In the following section, we present background on automatic pronunciation of print, intended to help the reader understand the difficulty of the problem. In Section 3, the process of letter–phoneme alignment is described since this turns out to be a necessary prior step for the deployment of PbA. In Section 4, we outline the principles of PbA in a little detail as this forms the essential background for this paper. The three dictionaries used in this work are described in Section 5. Section 6 details and compares the four approaches we have used to the fusion of the multiple heuristic scoring methods to produce a final decision, and Section 7 presents results obtained using these different approaches. Finally, Section 8 concludes with some discussion of the findings.

2. The problem of automatic pronunciation of print

English is notorious for the lack of regularity in its spelling-to-sound correspondence, which largely reflects the many complex historical influences on the spelling system [5–7]. Indeed, Abercrombie [8, p. 209] describes English orthography [2] as "... one of the least successful applications of the Roman alphabet". We use 26 letters in English orthography yet about 40–60 phonemes in specifying pronunciation. (See [9] for a specification of the phoneme symbols of the International Phonetic Alphabet.) It follows that the relation between letters and phonemes cannot be simply one-to-one. For instance, the letter *c* is pronounced /s/ in *cider* but /k/ in *cat*. On the other hand, the /k/ sound of *kitten* is written with a letter *k*. Nor is this lack of invariance between letters and phonemes the only problem. There is no strict correspondence between the number of letters and the number of phonemes in English words. Letter combinations (*ch*, *gh*, *ll*, *ea*) frequently act as a 'functional spelling unit' [10] signaling a single phoneme. Thus, the combination *ough* is pronounced /ʌf/ in *enough*, while *ph* is pronounced as the single phoneme /f/ in *phase*. However, *ph* in *uphill* is pronounced as two phonemes, /ph/.

Because functional spelling units are so common, there are usually fewer phonemes than letters in a word, but there are exceptions, e.g., *six* pronounced /sɪks/. Pronunciation can depend upon word class (e.g., *convict*,

[1] The phoneme is the minimal contrastive unit between sounds of a language in the sense of signaling a distinction between different words. Phonemes are written between slashes thus: //. Hence, the /t/ versus /d/ distinction between the words *tin* and *din* is phonemic in English, but the distinction between /p/ as in *pit* and /p/ as in *spit* is not. Even though the two /p/ sounds are acoustically rather different, this sort of acoustic difference is never used to distinguish between words of English.

[2] Orthography is the visual representation of language in the form of writing or printed text.

subject). English also has non-contiguous 'markings' [10] as, for instance, when the letter *e* is added to *mad* to give *made*, the pronunciation changes from /mad/ to /meɪd/. The final *e* is not sounded; rather it indicates that the vowel is lengthened or dipthongized. Such markings can be quite complex, or long-range, as when the suffix *y* is added to *photograph* or *telegraph* to yield *photography* or *telegraphy* respectively. English also contains many proper nouns (place names, surnames) which display idiosyncratic pronunciations, and loan words from other languages which conform to a different set of (partial) regularities. These further complicate the situation.

Given the complications described above, how is it possible to perform automatic phonemization at all? It is generally believed that the problem is largely soluble provided sufficient context is available. For example, the substring *ough* is pronounced /oʊ/ when its left context is *th* in the word *although*, /u/ when its left context is *thr* in the word *through*, and /ʌf/ when its left context is *en* in the word *enough*: In each case, the right context is the implicit word delimiter symbol. This idea of using context to disambiguate the inconsistencies of the orthographic to phonemic mapping led to the widespread use of context-dependent rewrite ("letter-to-sound") rules in TTS systems. These rule sets are manually compiled by expert linguists to reflect their knowledge and opinions of spelling-to-sound regularities in the language of interest. Well-known rule sets for English include those of Ainsworth [11], Elovitz et al. [12], Hunnicutt [13] and Divay and Vitale [14].

For a long time, the rule-based approach was considered more than adequate and there was a corresponding tendency to view the problem of automatic pronunciation generation as essentially solved. By contrast, machine learning or *data-driven* approaches—which attempt to infer pronunciations of unknown words from examples of known words and their pronunciations—were seen as strictly second-best and in great need of further development before they could compete. It is now abundantly clear, however, that this view of the superiority of rules over data-driven methods was over-optimistic, principally as a result of failure to test the rules on large enough sets of sufficiently demanding data [15,16]. The defining concept of rules—that capturing the regularities of letter–sound correspondence is sufficient for high-accuracy pronunciation—is flawed. As we have seen, inconsistencies are ubiquitous; they need to be processed as such and not merely forced inappropriately into an over-regularized framework. In the words of Daelemans, van den Bosch and Zavrel, "forgetting exceptions is harmful" [17]. Accordingly, we favour the data-driven approach of PbA, not only because it explicitly retains knowledge of exceptional pronunciations but also because it has consistently been the best-performing method in our work (e.g., [15]).

3. Letter–phoneme alignment

Many of the data-driven techniques for automatic phonemization, and PbA is no exception, require access to a dictionary of examples in which each word's spelling (in letters) and its pronunciation (in phonemes) have been aligned in one-to-one fashion [4]. The aligned dictionary serves as training data for machine learning approaches to automatic pronunciation, including as a knowledge base for analogical reasoning [16]. However, the lack of a consistent one-to-one mapping of letters to phonemes means that any such alignment can only be problematic. For instance, for words where there are different numbers of letters and phonemes, null symbols must be introduced. Consider the word *quay*, pronounced /ki/, for which a reasonable alignment might be

$$\begin{array}{cccc} q & u & a & y \\ | & | & | & | \\ k & - & i & - \end{array}$$

This word has fewer phonemes than letters, necessitating the insertion of two null phonemes in the pronunciation. These are entirely 'artificial' in that they play no role in specifying the pronunciation; their only purpose is to maintain the one-to-one correspondence between letters and phonemes. Yet it is not clear precisely where the nulls should be placed, since the following is also a reasonable alignment:

$$\begin{array}{cccc} q & u & a & y \\ | & | & | & | \\ - & k & - & i \end{array}$$

This illustrates the problem of functional spelling units, like *qu* → /k/. Unfortunately, any of the letters of the functional spelling unit could plausibly align with the corresponding phoneme, with the others corresponding to nulls, leading to a degree of indeterminacy.

More rarely, there are fewer letters than phonemes in a word of English. Examples are *six* (pronounced /sɪks/) and *sex* (pronounced /sɛks/) in which the single letter *x* maps to the two phonemes /ks/, so that null letters may have to be introduced to maintain a one-to-one mapping. As with null phonemes, the problem arises as to exactly where the nulls should be placed. Worse yet, both problems—null letters and null phonemes—can occur in the same word, as in the case of *axe*, pronounced /aks/, for which a reasonable alignment is

$$\begin{array}{cccc} a & x & - & e \\ | & | & | & | \\ a & k & s & - \end{array}$$

So, although there are the same numbers of letters and phonemes, we cannot assume a straightforward letter-to-phoneme alignment without nulls, a fact which defeats most simple approaches to alignment.

These examples illustrate that there is no canonically correct alignment of text and phonemes in every case, nor should we expect this, since the process is essentially a computational convenience lacking any sound linguistic or theoretical basis.

4. Principles of pronunciation by analogy

Pronunciation by analogy (PbA) is a data-driven technique for the automatic pronunciation of print, first proposed for TTS applications by Dedina and Nusbaum [18,19], hereafter D&N. It shares similarities with the artificial intelligence paradigms variously called case-based, memory-based or instance-based reasoning as applied to letter-to-phoneme translation [20–23]. These methods fit into the machine learning paradigm of 'lazy learning' [24] in that example data are remembered in their entirety. This has the advantages of minimising prior training and avoiding over-regularization errors caused by compressing the training data to the extent of removing information (e.g., as happens in neural network training, where regularities in the data are encoded into a set of connection weights and thresholds).

PbA exploits the phonological knowledge implicitly contained in a dictionary of words and their corresponding pronunciations. The underlying idea is that a pronunciation for an unknown word is derived by matching substrings of the input to substrings of known, lexical words, hypothesizing a partial pronunciation for each matched substring from the phonological knowledge, and assembling the partial pronunciations. Although initially it attracted little attention from workers in speech synthesis, several groups around the world are now trying to develop the approach as a back-up to dictionary matching [25–31].

The principles of PbA are best described with reference to the classical Pronounce program of D&N. This consists of four components: the lexical database (or dictionary), the matcher which compares the target input to all the words in the database, the pronunciation lattice (a data structure representing possible, or candidate, pronunciations), and the decision function, which selects the 'best' pronunciation among the set of possible ones. This selection is heuristic rather than being based on any theoretical model.

4.1. Pattern matching

The input word is first compared to words listed in the dictionary (*Webster's Pocket Dictionary* in D&N's work) and substrings common to both are identified. For a given dictionary entry, the process starts with the input string and the dictionary entry left-aligned. Substrings sharing contiguous, common letters in matching positions in the two strings are then found. Information about these matching letter substrings—and their corresponding phoneme substrings in the dictionary entry under consideration—is entered into the input string's pronunciation lattice as detailed below. This obviously requires the letters and phonemes of each word in the dictionary to have been previously aligned in one-to-one fashion as described in Section 3. The shorter of the two strings is then shifted right by one letter and the matching process repeated. This continues until the two strings are right-aligned (i.e., the number of right shifts is equal to the difference in length between the two strings).

D&N's pattern matching is, in our terms, 'partial'. That is, it starts with the left-most letter of the input string and of the current dictionary entry aligned and continues until the two are right-aligned. There seems to be no essential reason for starting and discontinuing matching at these points. That is, we could shift and match over the range of all possible overlaps. We call this 'full' as opposed to 'partial' matching. One conceivable objection to partial pattern matching is that some syllables or morphemes can act both as prefix and suffix (e.g., *someBODY* and *BODYguard*). A linguistic justification for the full method is that affixation is often implicated in the creation of new words. In this work, therefore, we have used full pattern matching throughout.

4.2. Pronunciation lattice

Matched substrings, together with their corresponding phonemic mappings as found in the dictionary, are used to build the pronunciation lattice for the input string. A node of the lattice represents a matched letter, L_i, at some position, i, in the input. The node is labeled with its position index i and with the phoneme which corresponds to L_i in the matched substring, P_{im} say, for the mth matched substring. An arc is placed from node i to node j if there is a matched substring starting with L_i and ending with L_j. The arc is labeled with the phonemes intermediate between P_{im} and P_{jm} in the phoneme part of the matched substring. Additionally, arcs are labeled with a 'frequency' count (see below) which is incremented by one each time that substring (with that pronunciation) is matched during the pass through the dictionary.

Fig. 1 shows an example pronunciation lattice for the word *longevity*. For clarity, the lattice has been simplified to show only a subset of the arcs—those contributing to the shortest path, since this is important to the decision on pronunciation (see below). The lattice was built by removing *longevity* from Webster's dictionary (see Section 5.2) and then matching it against the remaining 19,595 entries. This *leave one out* strategy is a very convenient way to derive pronunciations for all the words in a given dictionary, and thereby to assess

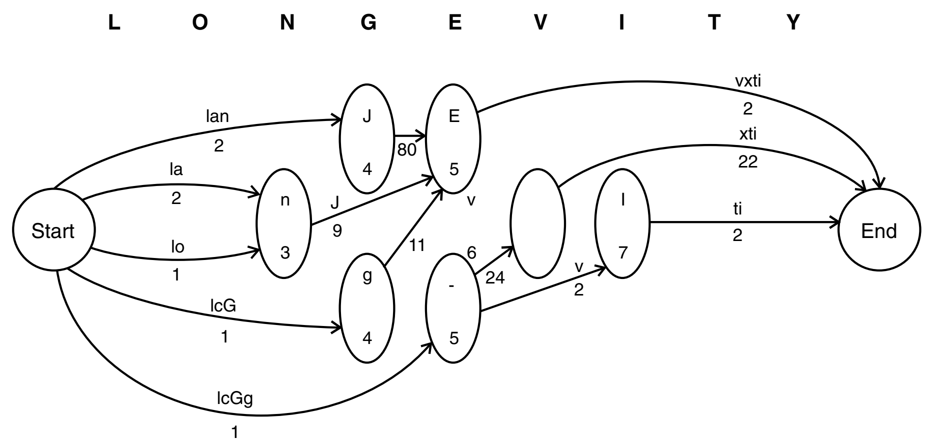

Fig. 1. Example pronunciation lattice for the word *longevity*. For simplicity, only arcs contributing to the shortest (length-3) paths are shown.

PbA performance; it is also known as *k*-fold cross-validation [32], where *k* is the size of the dictionary. The candidate pronunciations are not necessarily distinct. Different shortest paths can obviously correspond to the same phoneme string, as in this example.

4.3. Decision function

A possible pronunciation for the input string then corresponds to a complete path through its lattice, with the output string assembled by concatenating the phoneme labels on the nodes/arcs in the order that they are traversed. Different paths can, of course, correspond to the same pronunciation. Scoring of candidate pronunciations uses two heuristics in Pronounce:

1. If there is a unique shortest path, then the pronunciation corresponding to this path is taken as the output.
2. If there are tied shortest paths, then the pronunciation corresponding to the best scoring of these is taken as the output.

In D&N's original work, the score used in Heuristic 2 is the sum of arc 'frequencies' [3] obtained by counting the number of times the corresponding substring matches between the input and the entire dictionary. The scoring heuristics are one obvious dimension on which different versions of PbA can vary. In this work, we follow D&N in adopting Heuristic 1 throughout, so that when we refer to a *multistrategy* approach to PbA, we are referring to the use of several different scoring strategies for Heuristic 2.

[3] This is D&N's term, and is nothing to do with frequency of word usage/occurrence in written or spoken communication.

5. Dictionaries

In this work, we have used three different dictionaries as the knowledge base for the analogical generation of pronunciations. These dictionaries have different numbers of words and use different phoneme codings to represent pronunciations (which may also differ between dictionaries for what is nominally the same word).

5.1. Teachers Word Book

This database consists of 16,280 words from the *Teachers Word Book* (hereafter TWB) without homonyms. [4] TWB has been manually transcribed and aligned for research in TTS synthesis by McCulloch et al. [33, p. 294], who used it for training their NETspeak neural network. Although they do not say so in their paper, the transcriptions are British English, and some attempt has been made to anglicise spellings. The database is not publicly available and was supplied to us by the original authors, whose generosity is gratefully acknowledged. The phoneme inventory is of size 53. Like S&R's database (see below), TWB contains null phonemes but not null letters.

5.2. Webster's Pocket Dictionary

This database consists of the 20,008 words of the 1974 edition of *Webster's Pocket Dictionary*. It has been manually aligned by Sejnowski and Rosenberg [34], hereafter S&R, for training their NETtalk neural

[4] We exclude from our definition of homonym those word pairs which are spelled alike and pronounced alike but are different in meaning, such as *bank* (financial institution) and *bank* (in the sense of a bank of earth). These are considered to be the same word, and are in any case not separately listed in our dictionaries, which omit any definition of meaning.

network. At the time of this work, the database was freely available from http://www.speech.cs.cmu.edu/comp.speech for non-commercial use. The phoneme inventory is of size 51 which includes the null phoneme, as well as 'new' phonemes such as /K/, invented by S&R for the $x \rightarrow$ /ks/ correspondence, to avoid the use of null letters. Homographs (413 entries) were removed from the original NETtalk corpus to leave 19,596 entries. [5] Excluding homonyms keeps the conversion problem tractable. We did not want the same spelling to have different 'correct' pronunciations, otherwise we have to decide which to consider correct or accept any of them, but it is not clear how to make this decision.

5.3. British English Example Pronunciations

At the time of this work, the British English Example Pronunciation (BEEP) dictionary was publicly accessible from http://www.speech.cs.cmu.edu/comp.speech for non-commercial use. It is more typical of the size and content of the on-line dictionaries used for current speech technology applications than TWB or Webster's. BEEP was constructed by amalgamating several public domain dictionaries to yield a large composite. The complete version contains 257,033 words. There has been no strong quality control in constructing BEEP. Consequently, it contains several word entries (e.g., *abnegation*, *indissoluble*, *undiluted*) for which the pronunciations are clearly incorrect. Those that we discovered have been removed but we cannot guarantee to have found all errors. We also removed all homonyms as for the other two dictionaries. This gives 198,632 entries in all. The phoneme inventory is of size 45 including the null phoneme.

The version of BEEP archived at comp.speech is unaligned. We have aligned it automatically using our own algorithm [35]. Alignment results were slightly different according to the particular initialization used for the algorithm, which then iterates to improve the initial estimates of associations between letters and phonemes. In the work reported here, we used McCulloch et al.'s manual alignment of TWB as the initialization.

Our automatic alignment introduces null letters which are a problem for PbA, since unknown words in the input to a practical TTS system will never include null letters. To cope with this, we have removed words with null letters in their alignment, reducing the number of words to be tested from 198,632 to 178,041. This is an obvious simplification of the problem, but should nonetheless yield interesting results.

[5] This is very slightly different from the 19,594 word version of Webster's used by us in [30] in which we excluded the two one-letter words *I* and *a*.

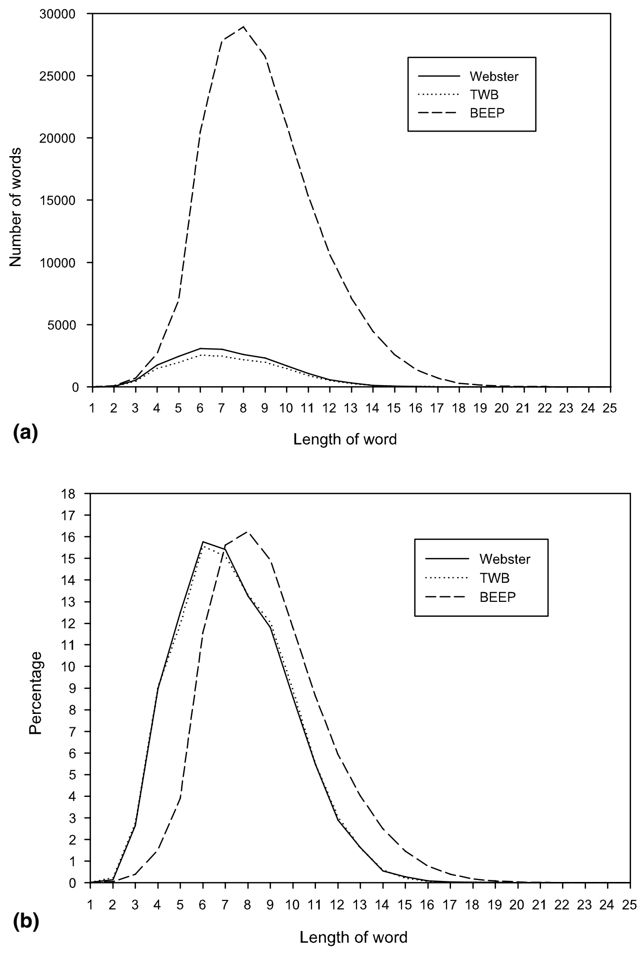

Fig. 2. Distribution of word lengths in the three dictionaries in terms of (a) numbers of words of a given length and (b) percentage of words of a given length.

5.4. Distribution of word lengths

Fig. 2 shows the distribution of word lengths for the three dictionaries. In Fig. 2(a), we show the distribution in terms of numbers of words; in Fig. 2(b), we show the distribution in terms of percentage of words. For Webster's dictionary, the average word length is 7.37 letters with a standard deviation of 2.43 letters. The longest word listed is *counterintelligence*. For TWB, the average word length is 7.38 letters also with a standard deviation of 2.43 letters. The longest word listed is *characteristically*. For BEEP, the average word length is 8.87 letters with a standard deviation of 2.58 letters. The longest word listed is *antidisestablishmentarian*.

6. Combining multiple scoring strategies

Following our earlier work [30], we have used five different scoring strategies as "soft experts", singly and in combination, for Heuristic 2 of the decision function (see Section 4.3). These are subsequently fused to produce a final decision. The chosen strategies are by no means exhaustive, nor do we make any claim that they represent the 'best' choices. Mostly, they were intuitively appealing heuristic measures that had different motivations in the hope that this would produce uncorrelated outputs, as this is often taken to be a prerequisite for successful information fusion. Also, we chose deliberately to include some very simple strategies indeed, probably over-simple, to see if they harmed performance.

6.1. Pronunciation candidates

Formally, the pronunciation lattice for a word after applying Heuristic 1 of Section 4.3 can be seen as a set of N candidate pronunciations (corresponding to tied, shortest paths) with some features:

$\mathscr{L}(W_i) = \{C_1, \ldots, C_j, \ldots, C_N\}$ is the lattice for the word W_i with $C_{j \in [1,N]}$ denoting the candidates.
C_j is a 3-tuple (F_j, D_j, P_j) where:
$F_j = \{f_1, \ldots, f_n\}$ represents the set of arc frequencies along the jth candidate path (length n).
$D_j = \{d_1, \ldots, d_k, \ldots, d_n\}$ represents the 'path structure', i.e., the difference of the position index (within the word) of the nodes at either end of the kth arc.
$P_j = \{p_1, \ldots, p_m, \ldots, p_l\}$ is the set of pronunciation candidates with p_m's from the set of phonemes and l is the length of the word.

In the example lattice for the word *longevity* shown earlier (Fig. 1), there are six shortest (length-3) paths and so six candidate pronunciations as tabulated in Table 1. (Since Webster's dictionary was used, the phoneme symbols in Fig. 1 and Table 1 are those of S&R.) In this example, the correct pronunciation is that corresponding to Candidates 4 and 6.

6.2. Scoring strategies

Five different strategies have been used as soft experts in deriving an overall pronunciation.

Strategy 1: this is the product of the arc frequencies (*PF*) along the shortest path.

Table 1
Candidate pronunciations for the word *longevity* whose (simplified) pronunciation lattice is shown in Fig. 1

Candidate	Pronunciation	Arc frequencies	Path structure
1	/lcGgEvxti/	{1, 11, 2}	{4, 1, 5}
2	/lcGg-vxti/	{1, 24, 22}	{5, 1, 4}
3	/lcGg-vIti/	{1, 2, 2}	{5, 2, 3}
4	/lanJEvxti/	{2, 9, 2}	{3, 2, 5}
5	/lonJEvxti/	{1, 9, 2}	{3, 2, 5}
6	/lanJEvxti/	{2, 80, 2}	{4, 1, 5}

The correct pronunciation is that corresponding to Candidates 4 and 6.

$$PF(C_j) = \prod_{i=1}^{n} f_i$$

The pronunciation scoring the *maximum PF*( ) value is considered the 'best' candidate on the basis of Strategy 1.

Strategy 2: the standard deviation of the values associated with the path structure (*SDPS*).

$$SDPS(C_j) = \frac{\sqrt{\sum_{i=1}^{n}(d_i - \bar{d})^2}}{n} \quad \text{where } \bar{d} = \frac{\sum_{i=1}^{n} d_i}{n}$$

In this case, the pronunciation scoring the *minimum SDPS*( ) value is considered the best candidate.

Strategy 3: the frequency of the same pronunciation (*FSP*), i.e., the number of occurrences of the same pronunciation (denoted cand) within the tied shortest paths.

$$FSP(C_j) = \text{cand}\{P_j \mid P_j = P_k\}$$
$$\text{with } j \neq k \text{ and } k \in [1, N]$$

The candidate scoring the *maximum FSP*( ) value is considered the best.

Strategy 4: the number of different symbols (*NDS*) between a pronunciation candidate C_j and the other candidates.

$$NDS(C_j) = \sum_{i=1}^{l} \sum_{k=1}^{N} \delta(P_{j,i}, P_{k,i})$$

where $\delta(\ )$ is the Kronecker delta, which is 1 if pronunciations P_j and P_k differ in position i and is 0 otherwise.

The candidate scoring the *minimum NDS*( ) value is considered the best.

Strategy 5: weak link (*WL*), i.e., the minimum of the arc frequencies.

$$WL(C_j) = \min_i \{f_i\}, \quad i \in [1, n]$$

The candidate scoring the *maximum WL*( ) value is considered the best.

6.3. Fusion methods

At this stage, we have somehow to combine the different scores to produce an overall decision. In so doing, however, we confront a well-recognized problem in information fusion, namely that the values produced by the different experts are incommensurate. We call this the *common currency* problem. [6] For example, for three of the strategies (*PF*, *FSP* and *WL*) a larger value

[6] The difficulty, of course, is that often there is no common currency.

indicates a better pronunciation whereas for the remainder (*SDPS* and *NDS*) a smaller value indicates a better pronunciation. Because it takes the product of already quite large values, Strategy 1 (*PF*) produces typically very large values whereas Strategy 3, the frequency of the same pronunciation *FSP*, is necessarily quite small.

We have chosen to compare four different fusion methods. Since we do not know the impact that the common currency problem will have on results in this specific application, our first method attempts to fuse the raw scores with only a minimal amount of prior processing. The second method addresses the common currency problem by allocating points to candidates according to their positions in per-expert rankings of scores. It is thus rank fusion based on Borda counting [36], and is the method used in our previous work [30]. Since this method discards the numerical values of the obtained scores (which could contain useful information) in making a decision on pronunciation, our next method allocates (non-integer) 'points' in a non-uniform fashion according to score value. The final method additionally weights the expert strategies according to their past performance.

Kittler et al. [37] have considered the relative merits of several combination rules from a theoretical and experimental perspective. The rules compared were sum, product, max, min and majority. These rules have the advantage of being simple and not requiring prior training. In this and subsequent studies [38,39], it was found that the product rule is superior to the sum rule when estimation errors are low, but the performance of the product rule deteriorates dramatically as estimation errors increase. We have used both sum and product rules here. Consideration was given to using modified product fusion [40], which truncates scores below a certain threshold to be equal to that threshold, but we decided against it because of the difficulty of determining the optimal threshold(s).

6.3.1. Fusion of raw scores

The basic idea of raw score fusion is to minimise the amount of prior processing of the scores, i.e., as far as possible we ignore the common currency problem. However, one mandatory requirement before raw scores can be fused is that all experts should produce scores having the same 'polarity'. That is, larger values should all correspond to *better* pronunciations (or vice versa). Further, when using the product rule, we need to avoid any particular score taking a zero value since this will give rise to a zero fused score irrespective of the values of the other scores. To satisfy these requirements, we make a simple transformation on those scores for which larger values correspond to *poorer* pronunciations (i.e., *SDPS* and *NDS*: Strategies 2 and 4 respectively). This transformation subtracts the obtained score for candidate j from the maximum obtained for all candidates and adds one (to avoid zero values), while scores for the other strategies (1, 3 and 5) are left unaltered. Hence, we have:

$$\begin{aligned}
&\text{Strategy 1}: \quad s_1(C_j) = PF(C_j)\\
&\text{Strategy 2}: \quad s_2(C_j) = \max_k SDPS(C_k) - SDPS(C_j) + 1\\
&\text{Strategy 3}: \quad s_3(C_j) = FSP(C_j)\\
&\text{Strategy 4}: \quad s_4(C_j) = \max_k NDS(C_k) - NDS(C_j) + 1\\
&\text{Strategy 5}: \quad s_5(C_j) = WL(C_j)
\end{aligned}$$

Using the sum rule, the fused score for a candidate pronunciation is simply taken as the sum of scores for each of the S strategies. Since we investigate all possible combinations of strategies, so that not all strategies are necessarily included, the fused score is

$$FS_{+}^{\text{raw}}(C_j) = \sum_{i=1}^{S} \delta_i s_i(C_j) \tag{1}$$

and for the product rule:

$$FS_{\times}^{\text{raw}}(C_j) = \prod_{i=1}^{S} (\delta_i s_i(C_j) + (1 - \delta_i)) \tag{2}$$

where δ_i is the Kronecker delta which is 1 if strategy i is included in the combined score and 0 otherwise. Finally, the pronunciation corresponding to the candidate which obtains the best final score is chosen.

6.3.2. Rank fusion based on Borda counting

Rank fusion solves the common currency problem by allocating points to candidates according to their position in a per-expert ranking. The number of points given to a candidate for expert scoring strategy i is inversely related to its rank order on the basis of s_i, 1 point for finishing last and the maximum of N points for the winner. This is the method of Borda counting described by Parker [36] (except that his description allocates zero points for finishing last and $N - 1$ points for the winner). Thus, the total number of points (T) awarded for each strategy is

$$T(N) = \sum_{r=1}^{N} r = \frac{N(N+1)}{2}$$

where N is the number of candidate pronunciations ($N = 6$ in our *longevity* example, so that $T(6) = 21$).

Let cand(R_i) express the number of candidates which have the rank R for the scoring strategy i so that cand(R_i) > 1 if there are ties, otherwise cand(R_i) = 1. Then $P(C_j, R_i)$, the number of points awarded to the candidate C_j thanks to its rank on the basis of strategy i, is

$$P(C_j, R_i) = \frac{\sum_{i=R_i}^{R_i+\text{cand}(R_i)-1} (N - i + 1)}{\text{cand}(R_i)} \tag{3}$$

For the sum rule, the fused score for a candidate pronunciation is simply taken as the sum of the different numbers of points won for each of the S strategies. Again, since not all strategies are necessarily included:

$$FS_{+}^{\text{rank}}(C_j) = \sum_{i=1}^{S} \delta_i P(C_j, R_i) \tag{4}$$

and for the product rule:

$$FS_{\times}^{\text{rank}}(C_j) = \prod_{i=1}^{S} (\delta_i P(C_j, R_i) + (1 - \delta_i)) \tag{5}$$

As with raw score fusion, the pronunciation corresponding to the candidate which obtains the best final score is chosen.

6.3.3. Rank fusion based on non-uniform values

According to Parker [36, p. 114], there is a "uniformity assumption … *[behind Borda counting which]* … is often flawed". This assumption is that "the distance between each candidate, once sorted, is the same", namely it is one. Parker suggests that "a better idea is … to assign simple non-uniform values to the ranked items". Of course, the very simplest way to do this is to use the raw scores themselves, as in Section 6.3.1, but this does not address the common currency problem. In this subsection, we consider a rank fusion scheme which allocates non-uniform values to ranked candidates.

As with raw score fusion, the first step is to transform scores *SDPS* and *NDS* for the 'minimizing' strategies (Strategies 2 and 4), by subtracting the obtained score for candidate j from the maximum obtained for all candidates and adding one to avoid zero values. At this stage, all scores s_i have the same polarity. We now allocate a value to each score as follows:

$$v(C_j, s_i) = \frac{s_i(C_j) - \min_k s_i(C_k) + 1}{\max_k s_i(C_k) - \min_k s_i(C_k) + 1} \tag{6}$$

which results in a range of values $0 < v(C_j, s_i) \leqslant 1$ for all strategies. Note that $v(C_j, s_i)$ can never be zero because for the maximising strategies 1, 3 and 5, *PF*, *FSP* and *WL* can never be less than one, whereas for the minimising strategies 2 and 4, the initial transformation means that s_i can never be less than one either.

For the sum rule, the fused score for a candidate pronunciation is simply taken as the sum of the non-uniform points for each of the S strategies:

$$FS_{+}^{\text{non-uni}}(C_j) = \sum_{i=1}^{S} \delta_i v(C_j, s_i) \tag{7}$$

and for the product rule:

$$FS_{\times}^{\text{non-uni}}(C_j) = \prod_{i=1}^{S} (\delta_i v(C_j, s_i) + (1 - \delta_i)) \tag{8}$$

As with the other fusion methods, the pronunciation corresponding to the candidate which obtains the best final score is chosen.

6.3.4. Rank fusion based on non-uniform values weighted by prior behavior

Parker [36] considers rank fusion schemes in which the prior behavior of soft experts (as measured by confusion matrices) is taken into account and shows that such information can improve performance. The basic idea used here is to use non-uniform $v(C_j, s_i)$ values as in Eq. (6), but to weight them according to the past performance of strategy i. Our estimate of the past performance is that obtained using the methods of the previous subsection. There is a certain amount of circularity in so doing, since in deriving a pronunciation for word w we are using weights obtained from performance on the whole dictionary (including word w). In principle, this circularity could be removed using k-fold cross-validation [7] but only at enormous computational expense. Since, however, each word makes only a minimal contribution to the weight set for the entire dictionary, the degree of circularity is negligible.

With this approach, the fused sum rule is

$$FS_{+}^{\text{weighted}}(C_j) = \sum_{i=1}^{S} \delta_i w_i v(C_j, s_i) \tag{9}$$

where w_i is the weight attached to strategy i according to its prior performance.

Obviously, there is no point to such weighting when using the product rule. In this case, the product $w_1 w_2 \cdots w_i \cdots w_S$ simply acts as a single common factor.

7. Results

Results were obtained for letter-to-phoneme conversion of all words of a dictionary using the leave-one (k-fold cross-validation) method mentioned in Section 6.1 for each of the three dictionaries, for each of the four fusion methods, and for all possible combinations of soft expert strategy. Since there are five such strategies, the number of possible combinations is $(2^5 - 1) = 31$. The various combinations are denoted as a 5-bit code where '1' at position i indicates that strategy i was included in the combination and a '0' indicates that it was not. Thus, as an example, the code 00010 indicates that Strategy 4 (*NDS*) was used alone. For compactness,

[7] For a dictionary of size k, there would be k different sets of weights, each obtained by leaving word w out of the evaluation of performance, $1 \leqslant w \leqslant k$. We would then derive a pronunciation for word w using the particular weight set that was computed leaving word w out.

we tabulate results for the best single strategy, the best combination overall, and for all five strategies used together (11111).

By 'best' in this context, we refer to word accuracy since this is a more demanding measure than symbol (phoneme) accuracy [15]. For a word to count as 'correct', *all* phonemes of a word must be correct, including the null phoneme, evaluated on a one-to-one basis. Thus, if the correct pronunciation for *make* is /meɪk_/, [8] then /m_eɪk/ is scored as incorrect. This is a deliberately strict way of scoring. As regards phoneme accuracy, this is again assessed on a one-to-one basis counting nulls as legitimate symbols. Hence, /m_eɪk/ scores just 25% phonemes correct as a pronunciation of *make*. As we will see, phoneme accuracy is a good but not perfect predictor of word accuracy, since there will be a dependence on how phoneme errors are distributed across words. Because the best single strategy result does not depend on the fusion method, it will appear multiple times in the following tabulations. This is done deliberately to allow easy comparison of results across conditions.

7.1. Results for fusion of raw scores

Table 2(a) and (b) shows the results obtained with the raw score fusion method using the sum rule and the product rule, respectively.

There is a wealth of intriguing information to be gleaned from these results. First, it is striking that Strategy 3 (the frequency of the same pronunciation, *FSP*) is consistently the best scoring single strategy across all the dictionaries. In earlier work on multi-strategy PbA using Webster's dictionary alone [30], we remarked with some surprise on the good performance of *FSP* since, given its simplicity, we had not expected it to perform as well as it obviously does. And whereas Strategy 1 is relatively popular in PbA (e.g., [19,28]), we are not aware that any other researchers have ever used Strategy 3. Next, there is a very consistent pattern of performance across the three dictionaries with the best figures for BEEP and the worst for Webster's. This pattern is not obviously and simply related either to the number of words in the dictionary or to the size of the phoneme inventory. Although BEEP uses the smallest number of phonemes (44), so making it relatively easier to produce 'correct' pronunciations, results are better for TWB than for Webster's in spite of the phoneme set being slightly larger (53 as against 51).

Turning next to the main issue of interest, does fusion of scores from the soft experts improve performance? Consider first the results obtained using the sum rule (Table 2(a)). For TWB and Webster's, there is an improvement in word accuracy between the single best strategy and the best combination: For TWB, 69.31% words correct increases to 70.26% for the combination 01111; for Webster's, 62.91% words correct increases to 63.63% for the combination 00111. Although these may look like marginal improvements, the relatively large number of words involved make such intuitions deceptive. Using a binomial test (see [30, pp. 212–213]), the improvements are actually significant ($z = 2.64$, $p < 0.01$ and $z = 2.09$, $p < 0.02$ respectively, one-tailed tests). For BEEP, however, there is no improvement from fusion using the sum rule, with the best 'combination' actually being the best single strategy (*FSP*), and with the all combinations condition 11111 doing particularly badly. It is very noticeable that Strategy 1 (product of arc frequencies, *PF*) has a detrimental effect on fusion. It is absent from the best combinations for all three dictionaries. In particular, when Strategy 1 is added to the 01111 best combination for TWB to produce the 11111 result in which all five soft experts participate, performance drops dramatically from 72.06% to 67.06%. [9]

Table 2
Results of raw score fusion for the three dictionaries using (a) sum rule and (b) product rule

	Best single	Best combination	All combinations
(a) Sum rule			
TWB	00100	01111	11111
Words (%)	69.31	70.26	67.06
Phonemes (%)	93.77	94.14	93.53
Webster's	00100	00111	11111
Words (%)	62.91	63.63	60.09
Phonemes (%)	91.63	92.11	91.23
BEEP	00100	00100	11111
Words (%)	86.01	86.01	77.94
Phonemes (%)	97.64	97.64	96.47
(b) Product rule			
TWB	00100	01111 & 00111	11111
Words (%)	69.31	70.90	70.65
Phonemes (%)	93.77	94.20	94.21
Webster's	00100	01111	11111
Words (%)	62.91	64.73	63.76
Phonemes (%)	91.63	92.25	92.06
BEEP	00100	01110	11111
Words (%)	86.01	86.66	82.99
Phonemes (%)	97.64	97.79	97.26

[8] Note that the diphthong /eɪ/ counts as a *single* IPA phoneme.

[9] We remind the reader that apparently quite small differences can be enormously significant for statistical tests with sample sizes of tens of thousands. Here, $|z| = 8.92$, and $p \sim 0$, meaning that p is smaller than the smallest number that can be computed in Microsoft Excel, for both one- and two-tailed tests.

Consider next the results obtained using the product rule (Table 2(b)). Here, fusion is uniformly beneficial with highly significant improvements for the best combination relative to the best single strategy for all three dictionaries. For example, the z value for BEEP (86.01% words correct increasing to 86.66%) is 7.91, $p \sim 0$. As for the sum rule, Strategy 1 is harmful; it never appears in the best combination and its inclusion in the 11111 condition generally reduces performance relative to this combination, although not so dramatically as with the sum rule.

Overall, the product rule is uniformly superior to the sum rule for raw score fusion. Even the most marginal improvement (TWB, best combination, increasing from 70.26% for sum rule to 70.90% for product rule) is significant with $z = 1.79$, $p < 0.05$.

As expected, phoneme score is a good but not perfect predictor of word score performance. Generally, better word scores are associated with better phoneme scores but there are minor exceptions. For instance, the fall in word accuracy (70.90% to 70.65%) in going from the 01111 condition for TWB and the product rule to the 11111 condition is reflected in a very small rise in phoneme score (94.20% to 94.21%).

In spite of being a popular heuristic in PbA, the poor showing of Strategy 1 when included in the raw score fusion schemes was not unexpected. We hypothesized in advance that (because it takes a product of already quite large values) *PF* would 'swamp' the fused score, as a manifestation of the common currency problem. Results confirm this expectation.

7.2. Results for rank fusion based on Borda counting

Table 3(a) and (b) shows the results obtained with the rank fusion method based on Borda counting using the sum rule and the product rule respectively.

The pattern of results across the three dictionaries seen with raw score fusion (i.e., best results for BEEP and poorest for Webster's) is replicated here. In this case, however, highly significant improvements ($p \sim 0$) are obtained using fusion relative to the best single strategy in all cases. The benefit in using Borda counting to solve the common currency problem is clear to see, with much improved results compared to raw score fusion as in the previous subsection. Again, Strategy 3 is uniformly the best performer but Strategy 1 is now able to play a full part, with Strategies 1 and 3 always participating in the best combination. Further, use of all five strategies no longer has a such a detrimental effect for TWB and Webster's, although the 11111 condition for BEEP is still very significantly poorer than the best combination. Although word accuracies for the product rule always very slightly exceed those for the sum rule, the differences are far from significant.

Table 3
Results of rank fusion based on Borda counting for the three dictionaries using (a) sum rule and (b) product rule

	Best single	Best combination	All combinations
(a) Sum rule			
TWB	00100	11100	11111
Words (%)	69.31	71.94	71.81
Phonemes (%)	93.77	94.35	94.39
Webster's	00100	11111	11111
Words (%)	62.91	65.34	65.34
Phonemes (%)	91.63	92.43	92.43
BEEP	00100	10100	11111
Words (%)	86.01	87.43	86.16
Phonemes (%)	97.64	97.85	97.72
(b) Product rule			
TWB	00100	11100	11111
Words (%)	69.31	*71.99*	71.76
Phonemes (%)	93.77	94.36	94.39
Webster's	00100	11111	11111
Words (%)	62.91	*65.35*	*65.35*
Phonemes (%)	91.63	92.40	92.40
BEEP	00100	10100	11111
Words (%)	86.01	*87.48*	86.11
Phonemes (%)	97.64	97.89	97.70

7.3. Results for rank fusion with non-uniform values

Table 4(a) and (b) shows the results obtained using the rank fusion method with non-uniform scores with the sum rule and product rule respectively.

The same pattern of results across the three dictionaries is seen here. Again, there is a performance advantage to using information fusion in this case. The best combination is very significantly better than the best single strategy ($p \sim 0$) for all three dictionaries and for both sum and product rules. For the sum rule, the fusion results are uniformly and significantly better than those obtained with raw score fusion (and the sum rule). For the product rule, however, the results are not significantly different from those obtained with raw score fusion (and the product rule). However, fusion results are consistently poorer than those of the previous subsection, where uniform values were assigned to the ranked candidates. Hence, it seems that the use of non-uniform values is better than raw score fusion (for the sum rule only) but has not improved on Borda counting. Why is this?

In effect, the ranking and counting transformation replaces ratio scale data by ordinal scale data [10] and so

[10] Ratio scale data measure the amount of some quantity on an absolute scale (real or integer) with a natural zero. It makes sense to say that $2x$ is twice as much as x for ratio data. Ordinal data are 'weaker' in that only differences between values are meaningful.

Table 4
Results of rank fusion with non-uniform values for the three dictionaries using (a) sum rule and (b) product rule

	Best single	Best combination	All combinations
(a) Sum rule			
TWB	00100	11111	11111
Words (%)	69.31	71.39	71.39
Phonemes (%)	93.77	94.31	94.31
Webster's	00100	00111	11111
Words (%)	62.91	64.99	64.81
Phonemes (%)	91.63	92.30	92.28
BEEP	00100	01110	11111
Words (%)	86.01	86.68	84.99
Phonemes (%)	97.64	97.80	97.55
(b) Product rule			
TWB	00100	01111	11111
Words (%)	69.31	70.88	70.42
Phonemes (%)	93.77	94.19	94.18
Webster's	00100	00111	11111
Words (%)	62.91	64.75	63.59
Phonemes (%)	91.63	92.26	92.02
BEEP	00100	01110	11111
Words (%)	86.01	86.67	82.93
Phonemes (%)	97.64	97.79	97.25

obliterates all information relating to absolute values. The good results for Borda counting seem to indicate that such a radical solution is needed to the common currency problem for the PbA application. If an incorrect pronunciation is to be replaced by a correct one, then our transformation from ratio to ordinal scales must be sufficiently powerful to affect their relative rankings. It seems that retaining 'reduced' information about absolute values is not as powerful in this respect as discarding it. Note, however, that some benefit is obtained from the information reduction, as the non-uniform values method does out-perform the raw scores method, if only for the sum rule.

7.4. Results for rank fusion with non-uniform weighted values

Here, the non-uniform values obtained from Eq. (6) for pronunciation candidate C_j using strategy i are weighted by w_i, $1 \leqslant i \leqslant S = 5$, when entered into the sum rule, Eq. (9). This approach is not relevant for the product rule since the product of the individual weights simply acts as a common factor scaling all candidate values by the same proportion. In this work, we have used the word accuracy of the individual strategies as the individual weights.

This procedure made almost no difference to results relative to the non-uniform fusion method of the previous subsection. Accordingly, we do not present any detailed results here. We conjecture that the performances of the individual experts are too even to provide appropriately differentiated weights for Eq. (9). We do not rule out the possibility of finding an effective set of weights in the future (e.g., by gradient descent), but we have not pursued it at this stage.

8. Discussion and conclusions

We have compared four different information fusion schemes for deciding among competing candidate pronunciations produced by five different soft experts in PbA. We find that all fusion schemes have advantages over the single best expert in terms of improved word accuracies. Even the simplest approach of raw score fusion (after a transformation to ensure that all scores had the same 'polarity') outperformed the single best expert. Best results overall were found for rank fusion of candidate scores using the product rule. These best results are shown in italics in Table 3(b). However, the better results for the product rule versus the sum rule did not reach statistical significance. Rank fusion solves the common currency problem in radical fashion by removing all information about absolute score values. Other approaches (using what we have called non-uniform scores) that aimed to retain some information about relative distance between scores performed less well, although still better than simple raw score fusion. We believe this shows the advantage of having a totally common currency, at least for this application.

Generally (and contrary to our earlier findings in [30] which used Webster's dictionary alone), there was no advantage to using all combinations of expert strategies. In most cases, this performed less well than the best combination of strategies. This raises the issue of how, in practice, a system designer might determine which strategies should be included and which should be excluded in the fusion. One possibility would be to predetermine the best combination for whatever is going to be used as the system dictionary, and then use this pre-determined combination for unknown words. Another possibility is to use all available strategies in conjunction with Alkoot and Kittler's modified product fusion [40], which attenuates the effect of poorly-scoring strategies although, as mentioned earlier, the problem is then faced of determining the optimal threshold.

Statistically, it could be argued that there are only 5 in 31 chances of getting a best single expert but $(31 - 5)$ chances in 31 of exceeding the best single expert with a best combination. Similarly, the 'all combinations' condition has only one chance to win. This somewhat calls into question the significance of our results when comparing the different combinations. However, the levels of significance obtained are, in most cases, so dramatic ($p \sim 0$) that we believe such considerations could have only a very minor effect on our findings and their interpretation.

We have observed a very consistent pattern of results across the three dictionaries used, with highest performance seen for BEEP rather than the other two dictionaries. This is probably a consequence of (a) BEEP using a smaller phoneme inventory of 44 symbols (cf. 53 for TWB and 51 for Webster's) and (b) the fact that BEEP contains many very similar words differing only in inflection (e.g., *cudgel*, *cudgeled*, *cudgeler*, *cudgelers*, *cudgeling*, *cudgelings*, *cudgelled*, *cudgelling*, *cudgels*). Both factors combine to make BEEP an easier dataset for letter-to-phoneme conversion. At this stage, however, it is unknown why TWB seems to be an easier dataset than Webster's in spite of being very similar in numbers of words and size of phoneme inventory. Current work is investigating this further.

References

[1] D.H. Klatt, Review of text-to-speech conversion for English, Journal of the Acoustical Society of America 82 (3) (1987) 737–793.

[2] T. Dutoit, Introduction to Text-to-Speech Synthesis, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[3] J.M. Lucassen, R.L. Mercer, An information theoretic approach to the automatic determination of phonemic baseforms, in: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'84, San Diego, CA, 1984, pp. 42.5.1–42.5.4.

[4] R.I. Damper, Self-learning and connectionist approaches to text–phoneme conversion, in: J. Levy, D. Bairaktaris, J. Bullinaria, P. Cairns (Eds.), Connectionist Models of Memory and Language, UCL Press, London, 1995, pp. 117–144.

[5] R.L. Venezky, A Study of English Spelling-to-Sound Correspondences on Historical Principles, Ann Arbor Press, Ann Arbor, MI, 1965.

[6] D.G. Scragg, A History of English Spelling, Manchester University Press, Manchester, UK, 1975.

[7] E. Carney, A Survey of English Spelling, Routledge, London, UK, 1994.

[8] D. Abercrombie, Extending the Roman alphabet: some orthographic experiments of the past four centuries, in: R.E. Asher, E. Henderson (Eds.), Towards a History of Phonetics, Edinburgh University Press, Edinburgh, UK, 1981, pp. 207–224.

[9] International Phonetic Association, Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet, Cambridge University Press, Cambridge, UK, 1999.

[10] R.L. Venezky, The Structure of English Orthography, Mouton, The Hague, The Netherlands, 1970.

[11] W.A. Ainsworth, A system for converting English text into speech, IEEE Transactions on Audio and Electroacoustics AU-21 (3) (1973) 288–290.

[12] H.S. Elovitz, R. Johnson, A. McHugh, J.E. Shore, Letter-to-sound rules for automatic translation of English text to phonetics, IEEE Transactions on Speech and Audio Processing ASSP-24 (6) (1976) 446–459.

[13] S. Hunnicutt, Phonological rules for a text-to-speech system, American Journal of Computational Linguistics, Microfiche 57 (1976) 1–72.

[14] M. Divay, A.J. Vitale, Algorithms for grapheme–phoneme translation for English and French: applications for database searches and speech synthesis, Computational Linguistics 23 (4) (1997) 495–523.

[15] R.I. Damper, Y. Marchand, M.J. Adamson, K. Gustafson, Evaluating the pronunciation component of text-to-speech systems for English: a performance comparison of different approaches, Computer Speech and Language 13 (2) (1999) 155–176.

[16] R.I. Damper, Learning about speech from data: beyond NETtalk, in: Data-Driven Methods in Speech Synthesis [41], pp. 1–25.

[17] W. Daelemans, A. van den Bosch, J. Zavrel, Forgetting exceptions is harmful in language learning, Machine Learning 34 (1–3) (1999) 11–43.

[18] M.J. Dedina, H.C. Nusbaum, PRONOUNCE: a program for pronunciation by analogy, Speech Research Laboratory Progress Report No. 12, Indiana University, Bloomington, IN, 1986.

[19] M.J. Dedina, H.C. Nusbaum, PRONOUNCE: a program for pronunciation by analogy, Computer Speech and Language 5 (1) (1991) 55–64.

[20] C.W. Stanfill, Memory-based reasoning applied to English pronunciation, in: Proceedings of 6th National Conference on Artificial Intelligence, AAAI-87, Seattle, WA, 1987, pp. 577–581.

[21] C.W. Stanfill, Learning to read: a memory-based model, in: Proceedings of Case-Based Reasoning Workshop, Clearwater Beach, FL, 1988, pp. 406–413.

[22] A.R. Golding, Pronouncing names by a combination of case-based and rule-based reasoning, PhD thesis, Stanford University, CA, 1991.

[23] A. van den Bosch, W. Daelemans, Data-oriented methods for grapheme-to-phoneme conversion, in: Proceedings of 6th Conference of the European Chapter of the Association for Computational Linguistics, Utrecht, The Netherlands, 1993, pp. 45–53.

[24] D.W. Aha, Lazy learning, Artificial Intelligence Review 11 (1–5) (1997) 7–10.

[25] K.P.H. Sullivan, R.I. Damper, Novel-word pronunciation: a cross-language study, Speech Communication 13 (3–4) (1993) 441–452.

[26] S. Federici, V. Pirrelli, F. Yvon, Advances in analogy-based learning: false friends and exceptional items in pronunciation by paradigm-driven analogy, in: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'95) Workshop on New Approaches to Learning for Natural Language Processing, Montreal, Canada, 1995, pp. 158–163.

[27] F. Yvon, Prononcer par Analogie: Motivations, Formalisations et Évaluations, PhD thesis, ENST, Paris, France, 1996.

[28] R.I. Damper, J.F.G. Eastmond, Pronunciation by analogy: impact of implementational choices on performance, Language and Speech 40 (1) (1997) 1–23.

[29] P.C. Bagshaw, Phonemic transcription by analogy in text-to-speech synthesis: novel word pronunciation and lexicon compression, Computer Speech and Language 12 (2) (1998) 119–142.

[30] Y. Marchand, R.I. Damper, A multistrategy approach to improving pronunciation by analogy, Computational Linguistics 26 (2) (2000) 195–219.

[31] K.P.H. Sullivan, Analogy, the corpus and pronunciation, in: Damper [41], pp. 45–70.

[32] S.M. Weiss, C.A. Kulikowski, Computer Systems that Learn, Morgan Kaufmann, San Mateo, CA, 1990.

[33] N. McCulloch, M. Bedworth, J. Bridle, NETspeak—a re-implementation of NETtalk, Computer Speech and Language 2 (3/4) (1987) 289–301.

[34] T.J. Sejnowski, C.R. Rosenberg, Parallel networks that learn to pronounce English text, Complex Systems 1 (1) (1987) 145–168.

[35] R.I. Damper, Y. Marchand, J.-D.S. Marsters, A. Bazin, Aligning letters and phonemes for speech synthesis, in: Proceedings of 5th International Speech Communication Association (ISCA) Workshop on Speech Synthesis, Pittsburgh, PA, 2004, pp. 209–214.

[36] J.R. Parker, Rank and response combination from confusion matrix data, Information Fusion 2 (2) (2001) 113–120.

[37] J. Kittler, M. Hatef, R.P.W. Duin, J. Matas, On combining classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (3) (1998) 226–239.

[38] F.M. Alkoot, J. Kittler, Experimental evaluation of expert fusion strategies, Pattern Recognition Letters 20 (11–13) (1999) 1361–1369.

[39] D.M.J. Tax, M. van Breuklen, R.W.P. Duin, J. Kittler, Combining multiple classifiers by averaging or multiplying? Pattern Recognition 33 (9) (2000) 1475–1485.

[40] F.M. Alkoot, J. Kittler, Modified product fusion, Pattern Recognition Letters 23 (8) (2002) 957–965.

[41] R.I. Damper (Ed.), Data-Driven Methods in Speech Synthesis, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001.