A Multistrategy Approach to Improving Pronunciation by Analogy

Yannick Marchand*
University of Southampton

Robert I. Damper†
University of Southampton

Pronunciation by analogy (PbA) is a data-driven method for relating letters to sound, with potential application to next-generation text-to-speech systems. This paper extends previous work on PbA in several directions. First, we have included "full" pattern matching between input letter string and dictionary entries, as well as including lexical stress in letter-to-phoneme conversion. Second, we have extended the method to phoneme-to-letter conversion. Third, and most important, we have experimented with multiple, different strategies for scoring the candidate pronunciations. Individual scores for each strategy are obtained on the basis of rank and either multiplied or summed to produce a final, overall score. Five strategies have been studied and results obtained from all 31 possible combinations. The two combination methods perform comparably, with the product rule only very marginally superior to the sum rule. Nonparametric statistical analysis reveals that performance improves as more strategies are included in the combination: this trend is very highly significant ($p \ll 0.0005$). Accordingly for letter-to-phoneme conversion, best results are obtained when all five strategies are combined: word accuracy is raised to 65.5% relative to 61.7% for our best previous result and 63.0% for the best-performing single strategy. These improvements are very highly significant ($p \sim 0$ and $p = 0.00011$ respectively). Similar results were found for phoneme-to-letter and letter-to-stress conversion, although the former was an easier problem for PbA than letter-to-phoneme conversion and the latter was harder. The main sources of error for the multistrategy approach are very similar to those for the best single strategy, and mostly involve vowel letters and phonemes.

1. Introduction

Text-to-phoneme conversion is a problem of some practical importance. Possibly the major application is speech synthesis from text, where we need to convert the text input (i.e., letter string) to something much closer to a representation of the corresponding sound sequence (e.g., phoneme string). A further important application is speech recognition, where we may wish to add a new word (specified by its spelling) to the vocabulary of a recognition system. This requires that the system has some idea of the "ideal" pronunciation—or phonemic **baseform** (Lucassen and Mercer 1984)—of the word. Also, in recognition we have a requirement to perform the inverse mapping, i.e., for conversion from phonemes to text. Perhaps the techniques employed for the forward mapping can also be applied "in reverse" for phoneme-to-text conversion. Yet another reason for being interested in the problem of automatic phonemization is

* Image, Speech and Intelligent Systems (ISIS) Research Group, Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK. E-mail: ym@ecs.soton.ac.uk
† Image, Speech and Intelligent Systems (ISIS) Research Group, Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK. E-mail: rid@ecs.soton.ac.uk

that (literate) humans are able to read aloud, so that systems that can pronounce print serve as models of human cognitive performance.

Modern text-to-speech (TTS) systems use lookup in a large dictionary or lexicon (we use the terms interchangeably) as the primary strategy to determine the pronunciation of input words. However, it is not possible to list exhaustively all the words of a language, so a secondary or backup strategy is required for the automatic phonemization of words not in the system dictionary. The latter are mostly (but not exclusively) proper names, acronyms, and neologisms. At this stage of our work, we concentrate on English and assume that any such missing words are dictionary-like with respect to their spelling and pronunciation, as will probably be the case for many neologisms.

Even if the missing words are dictionary-like, automatic determination of pronunciation is a hard problem for languages like English and French (van den Bosch et al. 1994). In fact, English is notorious for the lack of regularity in its spelling-to-sound correspondence. That is, it has a **deep** orthography (Coltheart 1978; Liberman et al. 1980; Sampson 1985) as opposed to the **shallow** orthography of, for example, Serbo-Croatian (Turvey, Feldman, and Lukatela 1984). To a large extent, this reflects the many complex historical influences on the spelling system (Venezky 1965; Scragg 1975; Carney 1994). Indeed, Abercrombie (1981, 209) describes English orthography as "one of the least successful applications of the Roman alphabet." We use 26 letters in English orthography yet about 45–55 phonemes in specifying pronunciation. 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 (Coltheart 1984)—or grapheme—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/. Usually, there are fewer phonemes than letters but there are exceptions, e.g., (*six*, /sIks/). Pronunciation can depend upon word class (e.g., *convict, subject*). English also has noncontiguous markings (Wijk 1966; Venezky 1970) as, for instance, when the letter *e* is added to (*mad*, /mad/) to make (*made*, /meId/), also spelled *maid*! 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. As a final comment, although not considered further here, English contains many proper nouns (place names, surnames) that display idiosyncratic pronunciations, and loan words from other languages that conform to a different set of (partial) regularities. These further complicate the problem.

This paper is concerned with an analogical approach to letter-to-sound conversion and related string rewriting problems. Specifically, we aim to improve the performance of pronunciation by analogy (PbA) by information fusion, an approach to automated reasoning that seeks to utilize multiple sources of information in reaching a decision—in this case, a decision about the pronunciation of a word. The remainder of this paper is organized as follows: In the next section, we contrast traditional rule-based and more modern data-driven approaches (e.g., analogical reasoning) to language processing tasks, such as text-to-phoneme conversion. In Section 3, we describe the original (PRONOUNCE) PbA system of Dedina and Nusbaum (1986) in some detail as this forms the basis for the later work. Section 4 reviews our own work in this area. Next, in Section 5, we make some motivating remarks about information fusion and its use in computational linguistics in general. In Section 6, we present in some detail the

multistrategy (or fusion) approach to PbA that enables us to obtain clear performance improvements, as described in Section 7. Finally, conclusions are drawn and directions for future studies are proposed in Section 8.

2. Rule-based and Data-driven Conversion

Given the problems 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 word delimiter symbol. In view of this, context-dependent rewrite rules have been a popular formalism for the backup pronunciation strategy in TTS systems. The form of the rules, strongly inspired by concepts from generative phonology (Chomsky and Halle 1968, 14), is:

$$A[B]C \rightarrow D \tag{1}$$

which states that the letter substring B with left context A and right context C receives the pronunciation (i.e., phoneme substring) D. Such rules can also be straightforwardly cast in the IF... THEN form commonly featured in high-level programming languages and employed in expert, knowledge-based systems technology. They also constitute a formal model of universal computation (Post 1943). Conventionally, these rules are specified by an expert linguist, conversant with the sound and spelling systems of the language of concern. Typical letter-to-sound rule sets are those described by Ainsworth (1973), McIlroy (1973), Elovitz et al. (1976), Hunnicutt (1976), and Divay and Vitale (1997).

Because of the complexities of English spelling-to-sound correspondence detailed in the previous section, more than one rule generally applies at each stage of transcription. The potential conflicts that arise are resolved by maintaining the rules in a set of sublists, grouped by (initial) letter and with each sublist ordered by specificity. Typically, the most specific rule is at the top and most general at the bottom. In the Elovitz et al. rules, for instance, transcription is a one-pass, left-to-right process. For the particular target letter (i.e., the initial letter of the substring currently under consideration), the appropriate sublist is searched from top to bottom until a match is found. This rule is then fired (i.e., the corresponding D substring is right-concatenated to the evolving output string), the linear search terminated, and the next untranscribed letter taken as the target. The last rule in each sublist is a context-independent default for the target letter, which is fired in the case that no other, more specific rule applies.

We refer to the mapping between B and D in Equation (1) as a **correspondence**. Given a set of correspondences, we can **align** text with its pronunciation. For example, consider the word (*make*, /melk/). A possible alignment (which uses the null phoneme—see below) is:

m	a	k	e
|	|	|	|
m	eI	k	-

It should be obvious, however, that it is all but impossible to specify a canonical set of correspondences for words of English on which all experts could agree. For instance, why should we use the single-letter correspondences *a* → /eI/, *k* → /k/, and *e* → /−/ as above, rather than the composite *ake* → /eIk/, which captures the noncontiguous marking by the final *e*? Of course, alignment and correspondences are at the heart of the rule-based methodology.

So, although the context-dependent rule formalism has been vastly influential—from both theoretical linguistic and practical system implementation perspectives—it does have its problems. From a practical point of view, the task of manually writing such a set of rules, deciding the rule order so as to resolve conflicts appropriately, maintaining the rules as mispronunciations are discovered, etc., is very considerable and requires an expert depth of knowledge of the specific language. For these reasons, and especially to ease the problem of creating a TTS system for a new language, more recent attention has focused on the application of automatic techniques based on machine learning from large corpora—see Damper (1995) for a comprehensive review, van den Bosch (1997) for more recent discussion, and Dietterich (1997) for an accessible review of the underpinning machine learning methodologies.

The rule-based approach has also been challenged from a more theoretical point of view. For instance, Jones (1996, 1) describes the goal of his book *Analogical Natural Language Processing* as:

> to challenge the currently predominant assumption in the field of natural language processing that the representation of language should be done within the rule-based paradigm alone. For historical reasons, this traditional position is largely the result of the influence of Chomsky and his efforts to define language in terms of formal mathematics.... The contrasting view, taken here, is that language is not something that can be described in a neat and tidy way.

This is also the perspective adopted here.

It is also conceivable that data-driven techniques can actually outperform traditional rules. However, this possibility is not usually given much credence. For instance, Divay and Vitale (1997) recently wrote: "To our knowledge, learning algorithms, although promising, have not (yet) reached the level of rule sets developed by humans" (p. 520). Dutoit (1997) takes this further, stating "such training-based strategies are often assumed to exhibit much more intelligence than they do in practice, as revealed by their poor transcription scores" (p. 115, note 14).

Pronunciation by analogy (PbA) is a data-driven technique for the automatic phonemization of text, originally proposed as a model of reading, e.g., by Glushko (1979) and Kay and Marcel (1981). It was first proposed for TTS applications over a decade ago by Dedina and Nusbaum (1986, 1991). See also the work of Byrd and Chodorow (1985), which considers computer-based pronunciation by analogy but does not mention the possible application to text-to-speech synthesis. As detailed by Damper (1995) and Damper and Eastmond (1997), PbA shares many similarities with the artificial intelligence paradigms variously called case-based, memory-based, or instance-based reasoning as applied to letter-to-phoneme conversion (Stanfill and Waltz 1986; Lehnert 1987; Stanfill 1987, 1988; Golding 1991; Golding and Rosenbloom 1991; van den Bosch and Daelemans 1993).

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 backup to dictionary matching.

In spite of opinions to the contrary expressed in the literature (see above), there is accumulating evidence that PbA easily outperforms linguistic rewrite rules (Damper and Eastmond 1996, 1997; Yvon 1996; Bagshaw 1998). More recently, Damper et al. (1999) conducted a careful performance evaluation of four techniques for letter-to-sound conversion: rules, backpropagation neural networks (Sejnowski and Rosenberg 1987; McCulloch, Bedworth, and Bridle 1987), PbA and the IB1-IG method based on information gain weighting (Daelemans, van den Bosch, and Weijters 1997; van den Bosch 1997). Results showed PbA to be the best of the techniques evaluated by a significant margin. Although one obviously cannot say from a limited evaluation with just three competitors that PbA is the best method available, it is clearly worthy of serious consideration and further development. This paper marks a stage of that development.

As a psychological (or theoretical) model, PbA is "seriously underspecified" so that the implementor wishing to use analogy within the pronunciation component of a TTS system "faces detailed choices which can only be resolved by trial and error" (Damper and Eastmond 1997, 1). The impact of implementational choices on performance has been studied by Sullivan and Damper (1993), Damper and Eastmond (1996, 1997), and Yvon (1996, 1997). One important dimension on which implementations vary is the strategy used to score the candidate pronunciations that PbA produces. By and large, these investigators have sought the *single* best pronunciation strategy. However, the range of choices is wide, so that some rather different implementations with different performance can be produced, yet these are all (in some sense) pronunciation by analogy. This raises the possibility, which forms the main focus of this paper, that different *multiple* PbA strategies—whose outputs are combined to give the final pronunciation—might be used to good effect. If the different strategies make different errors, then it is conceivable that such a multistrategy approach can produce a lower error rate than even the best single strategy. Indeed, it may be that a strategy with a poor performance by itself can make a positive overall contribution to high-accuracy pronunciation derivation when used in concert with other strategies. It can be viewed as a "specialist" within a committee of experts (e.g., Jacobs et al. 1991; Dietterich 1997): most often its opinion is invalid, but occasionally—for inputs within its sphere of expertise—it produces the correct answer when the other "generalist" strategies do not. There is currently much interest in the topic of **information fusion** in a wide variety of application settings. This work can be seen as a specific instance of information fusion.

3. Dedina and Nusbaum's System

The results reported here were obtained using an extended and improved version of PRONOUNCE, the Dedina and Nusbaum (D&N) system, which we now describe.

3.1 Principles

The basic PRONOUNCE system consists of four components: the lexical database; the matcher, which compares the target input to all the words in the database; the pronunciation lattice (a data structure representing possible pronunciations); and the decision function, which selects the "best" pronunciation among the set of possible ones. Reflecting PbA's origins as an empirical, psychological model, this selection is heuristic

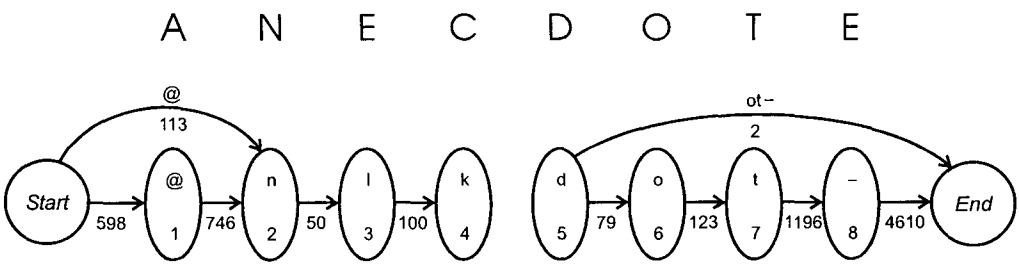

Figure 1
Simplified pronunciation lattice for the word *anecdote*. For clarity, only a subset of the arcs is shown. Full pattern matching is used as described in Section 3.2. Phoneme symbols are those employed by Sejnowski and Rosenberg.

rather than being based (like certain other approaches to automatic pronunciation) on any statistical model.

3.1.1 Pattern Matching. The input word is first compared to words listed in the lexicon (*Webster's Pocket Dictionary*) 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. (Note that this requires the letters and phonemes of each word in the lexicon to have been previously aligned in one-to-one fashion.) 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. This process can be alternatively seen as a matching between substrings of the incoming word "segmented in all possible ways" (Kay and Marcel 1981, 401) and the entries in the lexicon.

3.1.2 Pronunciation Lattice. Matched substrings, together with their corresponding phonemic mappings as found in the lexicon, 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 that 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 lexicon.

Figure 1 shows an example pronunciation lattice for the word *anecdote*. For clarity, the lattice has been simplified to show only a subset of the arcs. This word suffers from the so-called **silence problem** whereby PbA fails to produce any pronunciation, because there is no complete path through the lattice (see next page). In the case illustrated, there is no *cd* → /kd/ mapping in the dictionary other than in the word *anecdote* itself. Hence, in view of the leave-one-out testing strategy (see next page), there will never be an arc between nodes (/k/,4) and (/d/,5).

3.1.3 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 pronunciation uses two heuristics in PRONOUNCE. If there is a unique shortest path, then the pronunciation corresponding to this path is taken as the output. 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 is the sum of arc "frequencies" (Dedina and Nusbaum's term, and nothing to do with frequency of usage in written or spoken communication) obtained by counting the number of times the corresponding substring matches between the input and the entire lexicon. The scoring heuristics are one obvious dimension on which different versions of PbA can vary. In the following, when we refer to a multistrategy approach to PbA, it is principally the use of multiple scoring strategies which is at issue.

3.2 Appraisal

PRONOUNCE was evaluated on just 70 monosyllabic pseudowords—a subset of those previously used in reading studies by Glushko (1979). Such a test is largely irrelevant to TTS applications: the test set is not representative of general English, either in the small number of words used or their length. Also, D&N's claimed results on this pseudoword test set have proved impossible to replicate (Damper and Eastmond 1996, 1997; Yvon 1996; Bagshaw 1998). In addition, no consideration is given to the case where no complete path through the lattice exists (the silence problem mentioned earlier).

D&N's pattern matching (when building the pronunciation lattice) is a "partial" one. That is, as explained in section 3.1.1, the process starts with the leftmost letter of the input string and of the current dictionary entry aligned and continues until the two are right-aligned. They give (on page 59) the example of the input word *blope* matching to the lexical entry *sloping*. At the first iteration, the initial *b* of *blope* aligns with the initial *s* of *sloping*, and the common substring *lop* is extracted. The process terminates at the third iteration, when the final *e* of *blope* aligns with the final *g* of *sloping*: there are no common substrings in this case. 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—starting with the final *e* of *blope* aligned with the initial *s* of *sloping*, and terminating with the initial *b* of the former aligned with the final *g* of the latter. We call this "full" as opposed to "partial" matching. (Note that the simplified pronunciation lattice depicted in Figure 1 was obtained using full pattern matching.)

One conceivable objection to partial pattern matching is that some morphemes can act both as prefix and suffix (e.g., *someBODY* and *BODYguard*). From this point of view, full matching seems worth consideration. A linguistic justification for the full method is that affixation is often implicated in the creation of new words.

4. Previous Work and Extensions

In this section, we briefly review our previous work on a single-strategy approach to PbA (Damper and Eastmond 1996, 1997). The basic purpose of the earlier work was to reimplement D&N's system but to improve the scoring heuristic used to find the best path through the pronunciation lattice. To have a more realistic and relevant evaluation on a large corpus of real words, as opposed to a small set of pseudowords, we adopted the methodology of removing each word in turn from the lexicon and

deriving a pronunciation by analogy with the remaining words. In the terminology of machine learning, this is called **leave-one-out** or ***n*-fold cross validation** (Daelemans, van den Bosch, and Weijters 1997; van den Bosch 1997) where n is here the size of the dictionary. PbA has been used in this work to solve three string-mapping problems of importance in speech technology: letter-to-phoneme translation, phoneme-to-letter translation, and letter-to-stress conversion.

4.1 Lexical Database

The lexical database on which the analogy process is based is the 20,009 words of *Webster's Pocket Dictionary* (1974 edition), manually aligned by Sejnowski and Rosenberg (S&R) (1987) for training their NETtalk neural network. The database is publicly available via the World Wide Web from URL: ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries/. It has data arranged in columns:

aardvark	a-rdvark	1 <<<<> 2 <<
aback	xb@k-	0 > 1 <<
abacus	@bxkxs	1 < 0 > 0 <
abaft	xb@ft	0 > 1 <<
		etc.

Here the second column is the pronunciation, expressed in the phoneme symbols listed in S&R's Appendix A, pp. 161–162. (In this paper, we retain the use of S&R's phonetic symbols rather than transliterating to the symbols recommended by the International Phonetic Association. We do so to maintain consistency with S&R's publicly available lexicon.) The phoneme inventory is of size 52, and has the advantage (for computer implementation) that all symbols are single characters from the ASCII set. The "–" symbol is the null phoneme, introduced to give a strict one-to-one alignment between letters and phonemes to satisfy the training requirements of NETtalk. The third column encodes the syllable boundaries for the words and their corresponding stress patterns. According to S&R (Appendix A):

<	denotes	syllable boundary (right)
>	"	syllable boundary (left)
1	"	primary stress
2	"	secondary stress
0	"	tertiary stress

Stress is associated with vowel letters and arrows with consonants. The arrows point towards the stress nuclei and change direction at syllable boundaries. To this extent, "syllable boundary (right/left)" as above is a misnomer on the part of S&R: indeed, this information is not adequate by itself to place syllable boundaries, which we will denote "|". We can, however, infer four rules (or regular expressions) to identify syllable boundaries:

R1: [<>] ⇒ [< | >]
R2: [< digit] ⇒ [< | digit]
R3: [digit >] ⇒ [digit | >]
R4: [digit digit] ⇒ [digit | digit]

These have subsequently been confirmed as correct by Sejnowski and Rosenberg (personal communications).

Table 1 gives some examples of syllable boundaries and decoded "digit stress" patterns obtained using these rules (last row). By digit stress, we mean that the same

Table 1
Examples of stress and syllabification patterns.

Word	*abbreviate*	*abecedarian*	*actuarial*
Stress pattern	0 <> > 1 > 02 < <	2 > 0 > 0 > 1 < 00 <	2 <> 01 < 00 <
Syllabification	ab | bre | vi | ate	a | be | ce | dar | i | an	ac | tu | ar | i | al
Digit stress	00 | 111 | 00 | 222	2 | 00 | 00 | 111 | 0 | 00	22 | 00 | 11 | 0 | 00

stress-level code (one of 1, 2, or 0) is given to all letters—vowels and consonants—within a syllable. There are no ">" or "<" codes in the digit stress pattern.

For letter-to-phoneme conversion, homonyms (413 entries) and the two one-letter words *i* and *o* were removed from the original NETtalk corpus to leave 19,594 entries. (The one-letter word *a* is absent from Webster's dictionary.) We do not, of course, contest the importance of homonyms in general. Here, however, we focus on the (sub)problem of pronouncing isolated word forms. The assumption is that there will be another, extended process in the future that handles sentence-level pronunciation. Excluding homonyms keeps the problem tractable and means that we can have meaningful scores. 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. How do we make this decision? (Indeed, this was one of the problems in scoring Glushko's pseudowords as pronounced by D&N's system.) To apply the analogy method to phoneme-to-letter conversion, we have again used the NETtalk corpus. In this case, on the same logic as above, homophones (463 entries) and one-phoneme pronunciations (/A/ and /o/ in S&R's notation) were removed from this corpus. This left 19,544 pronunciations. Finally, we have used analogy to map letters to digit-stress patterns.

4.2 Best Preliminary Results

Preliminary results were obtained using a single scoring strategy in order to provide a baseline for comparison with the multistrategy approach to be described shortly. The best results (Table 2) for the three conversions—letter-to-phoneme, phoneme-to-letter, and letter-to-stress—were obtained by full pattern matching and using a weighted total product (TP) scoring. The latter is similar to Damper and Eastmond's TP score (Damper and Eastmond 1997), i.e., the sum of the product of the arc frequencies for all shortest paths giving the same pronunciation. In this case, however, each contribution to the total product is weighted by the number of letters associated with each arc.

Table 2
Best single-strategy results (% correct) for the three conversion problems studied: letter-to-phoneme, phoneme-to-letter, and letter-to-stress. These were obtained by full pattern matching and using a weighted total product (TP) scoring.

Letter-to-phoneme		Phoneme-to-letter		Letter-to-stress	
Words	Letters	Pronunciations	Phonemes	Words	Letters
61.7	91.6	74.4	94.6	54.6	75.0

The percentage of words in which both pronunciation and stress are correct was 41.8%. By use of a very simple strategy for silence avoidance, the results for letter-to-phoneme conversion were marginally increased from 61.7% to 61.9% words correct and from 91.6% to 91.8% phonemes correct. The strategy adopted was simply to add a (null-labeled) arc in the case that there was no complete path through the pronunciation lattice, and a single break occurred between adjacent nodes. This corresponds to concatenation of two otherwise complete word fragments. These best results should be compared with 60.7% words correct and 91.2% phonemes correct, as previously obtained by Damper and Eastmond (1997, Table 2).

5. Information Fusion in Computational Linguistics

In the introduction, we stated that our multistrategy approach is a special case of information (or data) fusion. What precisely is this? According to Hall and Llinas (1997, 7–8):

> The most fundamental characterization of ... fusion involves a hierarchical transformation between observed ... parameters (provided by multiple sources as input) and a decision or inference.

In principle, "fusion provides significant advantages over single source data" including "the statistical advantage gained by combining same-source data (e.g., obtaining an improved estimate ... via redundant observations)" (p. 6). However, dangers include "the attempt to combine accurate (i.e., good) data with inaccurate or biased data, especially if the uncertainties or variances of the data are unknown" (p. 8). Methods of information fusion include "voting methods, Bayesian inference, Dempster-Shafer's method, generalized evidence processing theory, and various *ad hoc* techniques" (Hall 1992, 135).

Clearly, the above characterization is very wide ranging. Consequently, fusion has been applied to a wide variety of pattern recognition and decision theoretic problems—using a plethora of theories, techniques, and tools—including some applications in computational linguistics (e.g., Brill and Wu 1998; van Halteren, Zavrel, and Daelemans 1998) and speech technology (e.g., Bowles and Damper 1989; Romary and Pierrel 1989). According to Abbott (1999, 290), "While the reasons [that] combining models works so well are not rigorously understood, there is ample evidence that improvements over single models are typical.... A strong case can be made for combining models across algorithm families as a means of providing *uncorrelated* output estimates." Our purpose in this paper is to study and exploit such fusion by model (or strategy) combination as a way of achieving performance gains in PbA.

6. Multiple Strategies for PbA

We have experimented with five different scoring strategies, used singly and in combination, in an attempt to improve the performance of PbA. 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 measures that had different motivations, chosen in the hope that this would produce uncorrelated outputs (see quote from Abbott above). Also, in view of the remarks of Hall and Llinas about the potential dangers of fusion/combination, we chose deliberately to include some very simple strategies indeed, to see if they harmed performance.

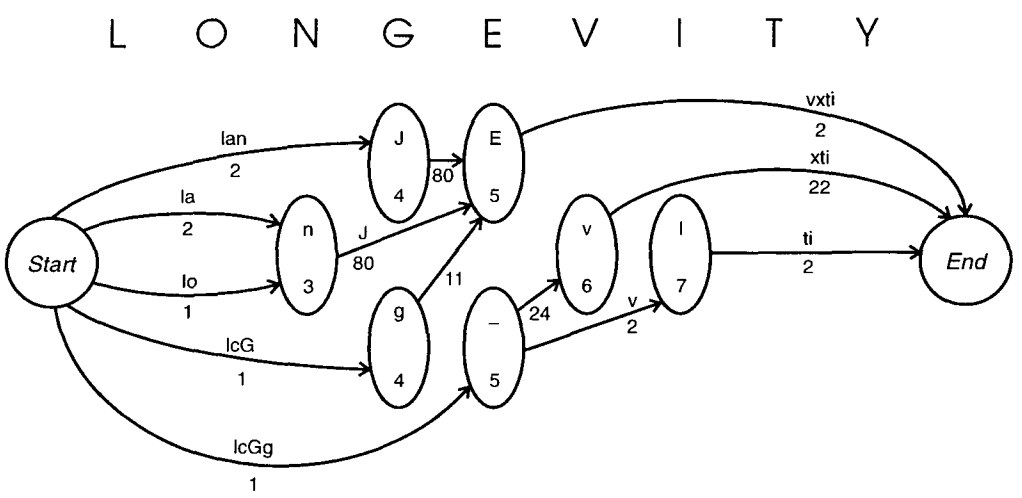

Figure 2
Example pronunciation lattice for the word *longevity*. For simplicity, only arcs contributing to the shortest (length-3) paths are shown and null arc labels are omitted. Phoneme symbols are those employed by Sejnowski and Rosenberg.

In the following, full pattern matching has been used exclusively.

6.1 Pronunciation Candidates

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

$\mathcal{L}(W_i) = \{C_1, \ldots, C_j, \ldots, C_N\}$ 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 (52 in our case) and l is the length of the pronunciation. (Within the NETtalk corpus, primarily because of the use of the null phoneme, words and their pronunciations all have the same length).

For example, for the pronunciation lattice given in Figure 2 for the word *longevity*, we have the six candidate pronunciations shown in Table 3, along with their arc frequencies and path structures. Of course, candidate pronunciations are not necessarily distinct: different shortest paths can obviously correspond to the same phoneme string. In this example, the correct pronunciation is that corresponding to Candidates 4 and 6.

6.2 Combining Scores

According to Hall and Llinas (1997, 8), "observational data may be combined or fused at a variety of levels." Since each of our strategies operates on the same basic data

Table 3
Candidate pronunciations for the word *longevity*. The correct pronunciation is that corresponding to Candidates 4 and 6.

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-vlti/	{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}

structure—the pronunciation lattice—the most obvious kind of fusion, and that employed here, is combination at the level of the final decision. In principle, fusion at this level is able to cope with the so-called common currency problem, whereby different sources of information produce incommensurate data of different types to which different physical units of measurement apply. Suppose, for instance, that combination is by a weighted summation in which the weights are learned from the data: in this case, the weights act to emphasize some measures while de-emphasizing others. Here, we use the **rank** of a pronunciation among the competing candidates as the basis of a weighting, or common currency.

From a computational point of view, the main idea is to attribute points to each candidate for each scoring strategy. The number of points given to a candidate for scoring strategy s_i is inversely related to its rank order on the basis of s_i. Thus, the total number of points (T) awarded for each strategy is:

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

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

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

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

Recently, Kittler et al. (1998) have considered the relative merits of several combination rules for decision-level fusion from a theoretical and experimental perspective. The rules compared were sum, product, max, min, and majority. Of these, the sum and product rules generally performed best. In view of this, these are the rules used here.

For the sum rule, the final score for a candidate pronunciation, $FS(C_j)$, is simply taken as the sum of the different numbers of points won for each of the S strategies.

Since not all strategies are necessarily included:

$$FS(C_j) = \sum_{i=1}^{S} \delta_{s_i} P(C_j, R_{s_i}) \tag{4}$$

and for the product rule:

$$FS(C_j) = \prod_{i=1}^{S} \delta_{s_i} P(C_j, R_{s_i}) + (1 - \delta_{s_i}) \tag{5}$$

where δ_{s_i} is the Kronecker delta, which is 1 if strategy s_i is included in the combined score and 0 otherwise. Finally, the pronunciation corresponding to the candidate that obtains the best final score is chosen.

6.3 Scoring Strategies

Five different strategies have been used in deriving an overall pronunciation. In the following, we list each strategy, define it, and give the result (in a table below the formal definition) of applying the strategy to the six candidate pronunciations of the example word *longevity* (Table 3). For each strategy, points are awarded as determined by Equation (3) and total 21 in accordance with Equation (2).

Strategy 1

This is the product of the arc frequencies (*PF*) along the shortest path.

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

The candidate scoring the maximum $PF(\,)$ value is given rank 1.

Candidate	1	2	3	4	5	6
Score, $PF(\,)$	22	528	4	36	18	320
Rank	4	1	6	3	5	2
Points	3	6	1	4	2	5

Strategy 2

The standard deviation of the values associated with the path structure (*SDPS*).

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

where

$$\overline{d} = \frac{\sum_{i=1}^{n} d_i}{n}$$

The candidate scoring the minimum *SDPS*( ) value is given rank 1.

Candidate	1	2	3	4	5	6
Score, *SDPS*( )	1.7	1.7	1.2	1.2	1.2	1.7
Rank	4	4	1	1	1	4
Points	2	2	5	5	5	2

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

$$FSP(C_j) = \text{cand}\{P_j | P_j = P_k\}$$

with

$$j \neq k \text{ and } k \in [1, N].$$

The candidate scoring the maximum *FSP*( ) value is given rank 1.

Candidate	1	2	3	4	5	6
Score, *FSP*( )	1	1	1	2	1	2
Rank	3	3	3	1	3	1
Points	2.5	2.5	2.5	5.5	2.5	5.5

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

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

where $\delta( )$ is the Kronecker delta, which is 1 if pronunciations P_j and P_k differ in position i and is 0 otherwise. Table 4 illustrates the computation of *NDS*( ) for Candidate 1 (/lcGgEvxti/).

The candidate scoring the minimum *NDS*( ) value is given rank 1.

Candidate	1	2	3	4	5	6
Score, *NDS*( )	12	14	18	13	14	13
Rank	1	4	6	2	4	2
Points	6	2.5	1	4.5	2.5	4.5

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

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

Table 4
Illustration of the computation of *NDS*( ) for the candidate pronunciation /lcGgEvxti/ of *longevity*. Phonemes which are not equal to those of the target pronunciation are entered in bold.

Candidate 1	l	c	G	g	E	v	x	t	i
	l	c	G	g	–	v	x	t	i
Other candidates	l	c	G	g	–	v	**I**	t	i
to be compared	l	**a**	**n**	**J**	E	v	x	t	i
with Candidate 1	l	**o**	**n**	**J**	E	v	x	t	i
	l	**a**	**n**	**J**	E	v	x	t	i
Differences at position *i*	0	3	3	3	2	0	1	0	0
Score, *NDS*( )					12				

The candidate scoring the maximum *WL*( ) value is given rank 1.

Candidate	1	2	3	4	5	6
Score, *WL*( )	1	1	1	2	1	2
Rank	3	3	3	1	3	1
Points	2.5	2.5	2.5	5.5	2.5	5.5

We now consider the results of using these five strategies for scoring the shortest paths both singly and combined in all possible combinations.

7. Results

In this section, we first detail some characteristics of the shortest paths through the pronunciation lattices (since these affect the attainable performance) before demonstrating that the combination strategy produces statistically significant improvements. It is largely immaterial if we use the sum or the product rule. Finally, the distribution of errors for the best-performing combination of scores is analyzed in order to set priorities for future research in improving PbA.

7.1 Characteristics of the Shortest Paths

Since we are focusing in this work on D&N's second heuristic (disambiguating tied shortest paths), it makes sense to investigate the limits set by tacit acceptance of D&N's first heuristic—which gives primacy to the shortest path. Table 5 shows some statistics related to the shortest paths for the three different conversion problems studied.

The minimal percentage indicates the lower bound on words-correct performance that obtains when the second heuristic is irrelevant, i.e., when all the shortest paths through the lattice give the identical, correct pronunciation. On the other hand, the maximal percentage indicates the upper bound that obtains when the second heuristic always chooses the correct candidate, i.e., there is at least one correct pronunciation among the shortest paths. Overall, these statistics indicate that there is considerable scope to improve the second heuristic, since the upper bound of 85.1% words correct for letter-to-phoneme conversion, for instance, is vastly superior to our previous best value of 61.9% and to the figure of 64.0% obtained by Yvon (1996) on the same lexicon using multiple unbounded overlapping chunks as the nodes of the pronunciation lattice. They also suggest (in line with our intuitions) that letter-to-phoneme conver-

Table 5
Statistics describing the occurrence of correct pronunciations among the shortest paths of the pronunciation lattices for all the words in the lexicon.

Type of Conversion	Minimal Percentage	Maximal Percentage	Mean Candidates per Word	Mean Good Candidates per Word
Letter-phoneme	15.3	85.1	7.9	2.2
Phoneme-letter	31.0	89.1	5.9	2.3
Letter-stress	11.6	78.2	7.5	2.0

sion is harder than phoneme-to-letter conversion, and that lexical stress assignment is harder still.

7.2 Results for Different Combinations of Strategy

We have obtained results for all possible combinations of the strategies, for each of the three mapping problems. Since there are five strategies, the number of combinations is $(2^5 - 1) = 31$. The various combinations are denoted as a five-bit code where 1 at position i indicates that strategy s_i was included in the combination (i.e., $\delta_{s_i} = 1$ in Equations (4) and (5)) and a 0 indicates that it was not. Thus, as an example, the code 00100 indicates that Strategy 3 (*FSP*) was used alone.

Table 6 gives an example of the use of the combination 11010 and product rule in deriving a pronunciation for the word *longevity* using the product rule of combination. The points that contribute to the final score are shown in bold. Note that the winner (Candidate 4) gives the correct pronunciation in this case. When the sum rule is used, the correct pronunciations (Candidates 4 and 6) tie with a final score of 13.5.

Table 7 shows the results of letter-to-phoneme conversion for the 31 possible combinations of scoring strategy using both the product and sum rules. The average across the 31 combinations was 62.42% words correct for the product rule compared to 62.30% for the sum rule. This difference is not significant (on the basis of Equation (6) below). Since the product rule gave numerically higher values, however, we continue to use it for the remainder of this paper.

Tables 8, 9, and 10 show the results obtained with all possible combinations of strategies for the three conversion problems using the product rule. Consider first the results for letter-to-phoneme conversion (Table 8). The last two columns show the rank according to the number of strategies included in the final score (Rank(*C*)) and the rank according to word accuracy (Rank(*W*)). Let us hypothesize that these two

Table 6
Example of multistrategy scoring for the word *longevity* using the 11010 combination and product rule.

Candidate	*PF*	*SDPS*	*FSP*	*NDS*	*WL*	Final Score
1	**3**	**2**	2.5	**6**	2.5	36
2	**6**	**2**	2.5	**2.5**	2.5	30
3	**1**	**5**	2.5	**1**	2.5	5
4	**4**	**5**	5.5	**4.5**	5.5	90
5	**2**	**5**	2.5	**2.5**	2.5	25
6	**5**	**2**	5.5	**4.5**	5.5	45

Winner: Candidate 4 (/lanJEvxti/)

Table 7
Results of letter-to-phoneme conversion for the 31 possible combinations of scoring strategy using the product and sum rules.

	Product rule		Sum rule	
Combination	Words (%)	Phonemes (%)	Words (%)	Phonemes (%)
00001	57.7	90.8	57.7	90.5
00010	61.3	91.7	61.3	91.7
00011	63.3	92.1	63.2	92.1
00100	63.0	91.7	63.0	91.7
00101	65.3	92.2	65.1	92.2
00110	63.2	91.9	63.1	91.9
00111	64.9	92.2	65.0	92.3
01000	48.2	88.4	48.2	88.4
01001	56.7	90.4	57.0	90.5
01010	61.5	91.7	61.5	91.7
01011	63.2	92.0	63.1	91.0
01100	62.7	91.6	62.8	91.6
01101	64.7	92.1	64.6	92.1
01110	63.0	91.9	63.0	91.9
01111	64.7	92.2	64.9	92.2
10000	59.2	91.0	59.2	91.0
10001	59.0	91.2	59.1	91.2
10010	63.6	92.1	63.6	92.1
10011	63.6	92.2	63.4	92.1
10100	65.2	92.2	65.1	92.2
10101	65.3	92.3	64.9	92.2
10110	64.8	92.2	64.8	92.3
10111	65.4	92.4	65.4	92.4
11000	58.6	91.1	58.6	91.1
11001	59.0	91.2	59.2	91.2
11010	63.6	92.2	63.6	92.2
11011	63.4	92.2	63.2	92.1
11100	65.4	92.3	65.3	92.3
11101	64.9	92.3	64.7	92.2
11110	65.2	92.3	65.1	92.3
11111	65.5	92.4	65.5	92.4

ranks are not positively correlated. That is, our null hypothesis is that performance (in terms of word accuracy) does *not* increase as more scoring strategies s_i for candidate pronunciation C_j are included in the final score $FS(C_j)$. However, the Spearman rank correlation coefficient r_s (Siegel 1956, 202–213) is computed here as 0.6657, with degrees of freedom $df = (31 - 2) = 29$. For $df \geq 10$, the significance of this result can be tested as:

$$t = r_s \sqrt{\frac{df}{1 - r_s^2}} = 4.8041$$

This value is very highly significant ($p \ll 0.0005$, one-tailed test). Hence, we reject the null hypothesis and conclude that performance improves as more scores are included in the combination. Note that this test is nonparametric, and makes a minimum of assumptions about the data—only that they are ordinal and so can be meaningfully ranked.

Table 8
Results of letter-to-phoneme conversion for the 31 possible combinations of scoring strategy using the product rule. Rank(C) is the rank of the result according to the number of strategies (in the range 1 to 5) included in the final score. Rank(*W*) is the rank of the result according to word accuracy. The Spearman rank correlation coefficient r_s is 0.6657, which is very highly significant.

Combination	Words (%)	Phonemes (%)	Rank(C)	Rank(*W*)
00001	57.7	90.8	29	29
00010	61.3	91.7	29	24
00011	63.3	92.1	21.5	17
00100	63.0	91.7	29	20.5
00101	65.3	92.2	21.5	4.5
00110	63.2	91.9	21.5	18.5
00111	64.9	92.2	11.5	8.5
01000	48.2	88.4	29	31
01001	56.7	90.4	21.5	30
01010	61.5	91.7	21.5	23
01011	63.2	92.0	11.5	18.5
01100	62.7	91.6	21.5	22
01101	64.7	92.1	11.5	11.5
01110	63.0	91.9	11.5	20.5
01111	64.7	92.2	4	11.5
10000	59.2	91.0	29	25
10001	59.0	91.2	21.5	26.5
10010	63.6	92.1	21.5	14
10011	63.6	92.2	11.5	14
10100	65.2	92.2	21.5	6.5
10101	65.3	92.3	11.5	4.5
10110	64.8	92.2	11.5	10
10111	65.4	92.4	11.5	2.5
11000	58.6	91.1	21.5	28
11001	59.0	91.2	11.5	26.5
11010	63.6	92.2	11.5	14
11011	63.4	92.2	4	16
11100	65.4	92.3	11.5	2.5
11101	64.9	92.3	4	8.5
11110	65.2	92.3	4	6.5
11111	65.5	92.4	1	1

Having shown that there is a very highly significant positive correlation between the number of strategies deployed and the obtained word accuracy, we next ask if the obtained improvement is significant. (This is to take account of the possibility that the difference between two combination strategies ranked at positions i and $(i + k)$ is not significant.) To answer this, we note that only two outcomes are possible for the translation of each word: either the pronunciation is correct or it is not. Thus, the sampling distribution of the word accuracies listed in the second column of Table 8 is binomial and, hence, we can use a binomial test (Siegel 1956, 36–42) to determine the significance of differences between them. Since the number of trials (i.e., word translations) is very large (~20,000), we can use the normal approximation to the binomial distribution.

Let us first ask if the best letter-to-phoneme conversion result here (65.5% word accuracy for combination 11111) is significantly better than the previous best, preliminary

value of 61.7%. The appropriate statistic is (Siegel 1956, 41):

$$z = \frac{(x \pm 0.5) - NP}{\sqrt{NPQ}} \tag{6}$$

where $N = 19,594$ words, $P = 0.617$, $Q = (1 - P) = 0.383$, $x = 19,594 \times 0.655$ and the ± 0.5 term (correction for the fact that the binomial distribution is discrete while the normal distribution is continuous) can be ignored, giving $z = 10.9$. The (one-tailed) probability that this value could have been obtained by chance is effectively zero (untabulated in Siegel's Table A [p. 247]). In fact, the critical value for the 1% significance level is $z = 2.33$, which equates to a word accuracy of approximately 64.7%. It follows that even the best single-strategy result (63.0% for combination 00100 using Strategy 3 only) is significantly poorer than the multistrategy result using all five scoring strategies.

Actually, given its simplicity, it is remarkable that Strategy 3 (frequency of the same pronunciation, *FSP*) used alone performs as well as it does. It was included partly to test the effect of including what were felt to be oversimplistic strategies! Yet it is superior to the previous best result of 61.7% using the weighted TP score, and the superiority is very highly significant ($z = 3.7$, $p = 0.00011$). In fact, for all three mapping problems, Strategies 1 (*PF*) and 3 are always implicated in results of rank less than 3, indicating their importance in obtaining high performance.

Turning to phoneme-to-letter conversion (Table 9), the Spearman rank correlation coefficient was 0.6375, which again is very highly significant ($t = 4.456$, $p \ll 0.0005$, $df = 29$, one-tailed test). Hence, as before, performance improves as more scoring strategies are deployed. The critical z value of 2.33 for the 1% significance level equates to a word accuracy of 74.7% relative to the best obtained word accuracy of 75.4% for combination 10101. Hence, the best result is significantly better than either the previous best value (74.4%) or the best single-strategy result (73.5% for combination 10000).

For letter-to-stress conversion (Table 10), the Spearman rank correlation coefficient was 0.7411 ($t = 5.944$, $p \ll 0.0005$, $df = 29$, one-tailed test) so that, once again, performance improves as more scoring strategies are deployed. The critical z value of 2.33 for the 1% significance level equates to a word accuracy of 58.0% relative to the best obtained word accuracy of 58.8% for combination 11100. Hence, the best result is significantly better than either the previous best value (54.6%) or the best single-strategy result (53.4% for combination 00100).

Finally, the percentage of words in which both pronunciation and stress are correct increases from 41.8% to 46.3%.

7.3 Analysis of Errors

Table 11 identifies the main sources of error for letter-to-phoneme conversion using the 11111 combination strategy. Table 11(a) indicates, in rank order, the 10 letters in the input that were most often mapped to an incorrect phoneme. The commonest problem is mispronunciation of letter *e*, which produces 21.2% of the total errors. To some extent, this is a natural consequence of the high frequency of this letter in English: as indicated in the Proportion column, letter *e* accounts for 11.0% of the total corpus. Even so, the ratio of errors to occurrences is almost 2, while it actually exceeds 2 for letters *a* and *o*. It is clear, as other workers have found, that the vowel letters are vastly more difficult to translate than are the consonant letters.

Table 11(b) ranks the 10 commonest incorrect phonemes in the system's output. The schwa vowel accounts for 20.8% of errors in this case. Again, this partially reflects the extremely common occurrence of this phoneme.

Table 9
Results of phoneme-to-letter conversion for the 31 possible combinations of scoring strategy using the product rule. Rank(C) is the rank of the result according to the number of strategies (in the range 1 to 5) included in the final score. Rank(W) is the rank of the result according to word accuracy. The Spearman rank correlation coefficient r_s is 0.6375, which is very highly significant.

Combinations	Words (%)	Phonemes (%)	Rank(C)	Rank(W)
00001	69.1	93.6	29	29
00010	71.6	94.2	29	27
00011	73.3	94.4	21.5	18
00100	72.2	94.2	29	22.5
00101	74.0	94.5	21.5	13
00110	72.3	94.2	21.5	21
00111	73.9	94.5	11.5	14
01000	61.6	92.1	29	31
01001	68.2	93.4	21.5	30
01010	71.5	94.1	21.5	28
01011	73.1	94.4	11.5	19
01100	72.0	94.1	21.5	24
01101	73.7	94.4	11.5	15.5
01110	72.2	94.2	11.5	22.5
01111	73.7	94.4	4	15.5
10000	73.5	94.4	29	17
10001	72.7	94.3	21.5	20
10010	74.3	94.6	21.5	10.5
10011	74.7	94.7	11.5	8
10100	75.1	94.7	21.5	2.5
10101	75.4	94.7	11.5	1
10110	74.8	94.6	11.5	6
10111	75.1	94.7	4	2.5
11000	71.7	94.0	21.5	26
11001	71.8	94.1	11.5	25
11010	74.3	94.6	11.5	10.5
11011	74.2	94.5	4	12
11100	75.0	94.6	11.5	4.5
11101	74.7	94.6	4	8
11110	74.7	94.6	4	8
11111	75.0	94.7	1	4.5

Finally, Table 11(c) shows the 10 phonemes in the correct pronunciation that most often received a wrong translation. These are the same 10 phonemes as for Table 11(b), but in slightly different rank order. Once more, it is clear that vowel errors vastly outnumber consonant errors overall. The null phoneme is also problematic.

This pattern of errors is very close to that for the preliminary results. We have exactly the same 10 main letters/phonemes responsible for errors in each column with only minor changes in their rank order. These similarities suggest that these particular errors are persistent—even structural—and will cause problems for other translation schemes as well as PbA.

8. Conclusions

We have extended previous work in pronunciation by analogy (PbA) principally by experimenting with multiple strategies for producing pronunciations. The combina-

Table 10
Results of letter-to-stress conversion for the 31 possible combinations of scoring strategy using the product rule. Rank(C) is the rank of the result according to the number of strategies (in the range 1 to 5) included in the final score. Rank(*W*) is the rank of the result according to word accuracy. The Spearman rank correlation coefficient r_s is 0.7411, which is very highly significant.

Combinations	Words (%)	Phonemes (%)	Rank(*W*)	Rank(*W*)
00001	52.7	74.9	29	26
00010	49.2	74.4	29	30
00011	53.6	75.7	21.5	23.5
00100	53.4	73.9	29	25
00101	57.7	76.3	21.5	4
00110	53.8	74.8	21.5	21.5
00111	56.2	75.9	11.5	11
01000	39.4	67.5	29	31
01001	52.2	74.4	21.5	27
01010	50.5	74.5	21.5	29
01011	53.9	75.8	11.5	20
01100	53.8	74.2	21.5	21.5
01101	57.1	76.1	11.5	7.5
01110	54.2	75.0	11.5	18
01111	56.2	75.9	4	11
10000	51.6	73.7	29	28
10001	54.1	75.3	21.5	19
10010	53.6	75.4	11.5	23.5
10011	55.2	76.0	21.5	14
10100	57.4	75.9	21.5	6
10101	58.2	76.6	11.5	3
10110	56.2	75.8	11.5	11
10111	57.1	76.3	4	7.5
11000	54.5	75.9	21.5	17
11001	54.6	76.1	11.5	16
11010	55.0	76.3	11.5	15
11011	55.8	76.7	4	13
11100	58.8	77.0	11.5	1
11101	58.6	77.2	4	2
11110	57.0	76.4	4	9
11111	57.5	76.7	1	5

tion of different scoring strategies for the shortest paths in the pronunciation lattice has been shown to deliver statistically significant improvements: a best result of 65.5% words correct has been obtained for letter-to-phoneme conversion using approximately 20,000 manually aligned words of *Webster's Pocket Dictionary*. This compares with a figure of 63.0% for the best-performing single-scoring strategy (frequency of the same pronunciation, *FSP*) and 61.7% for our best preliminary result. Examination of the pronunciation lattices, however, reveals that the upper bound on performance of a method based on selecting among shortest paths is 85.1%, so that there is scope for further improvement yet.

The way of combining scores is simply by summation or multiplication of a points score awarded on the basis of a pronunciation's rank. The product rule of combination is found to perform only very marginally better than the sum rule: the difference is not significant. We have not yet expended much effort in determining exactly which scores should be used. To this end, a preliminary analysis of errors has been done. This

Table 11
Main factors responsible for errors in letter-to-phoneme mapping.

(a) Letters most often mapped to an incorrect phoneme

Letter	Errors (%)	Proportion (%)
e	21.2	11.0
a	20.9	9.0
o	16.1	6.9
i	14.1	8.8
u	7.0	3.8
r	3.8	7.5
l	2.7	5.5
s	2.7	5.3
n	2.5	6.8
t	1.8	7.7

(b) Phonemes most commonly wrong in the output

Phoneme	Errors (%)	Proportion (%)
/x/	20.8	8.0
/I/	9.6	5.0
/-/	7.7	14.7
/E/	7.5	2.5
/@/	7.2	2.8
/a/	5.4	1.9
/i/	4.8	3.0
/e/	4.1	1.8
/o/	3.5	1.5
/A/	3.0	1.1

(c) Phonemes of the correct translation most commonly mispronounced

Phoneme	Errors (%)	Proportion (%)
/x/	22.4	8.1
/-/	9.9	14.8
/I/	7.7	4.8
/a/	6.3	1.9
/E/	6.0	2.4
/i/	5.7	3.1
/@/	5.3	2.7
/e/	4.2	1.8
/o/	3.6	1.5
/A/	3.5	1.2

reveals that (in common with other approaches to letter-to-sound conversion, such as linguistic rules), the translation of vowel letters is especially problematic. Future work should therefore attempt to find scoring methods and combination techniques that deal effectively with the vowels.

We have also studied the related problems of mapping phonemes to letters and letters to lexical stress, which are also important in speech technology. Our results show that the former problem is easier than letter-to-phoneme conversion while the latter is harder—at least, when attacked by the method of analogy. Once again, however, the multistrategy approach has the potential to deliver significant performance gains.

Acknowledgments

This work was funded by the UK Economic and Social Research Council via research grant R000235487: "Speech Synthesis by Analogy."

References

Abbott, Dean W. 1999. Combining models to improve classifier accuracy and robustness. In *Proceedings of Second International Conference on Information Fusion, Fusion '99*, Volume I, pages 289–295, Sunnyvale, CA.

Abercrombie, David. 1981. Extending the Roman alphabet: Some orthographic experiments of the past four centuries. In Ronald E. Asher and Eugénie J. A. Henderson, editors, *Towards a History of Phonetics*, pages 207–224. Edinburgh University Press, Edinburgh, UK.

Ainsworth, William A. 1973. A system for converting English text into speech. *IEEE Transactions on Audio and Electroacoustics AU-21*, pages 288–290.

Bagshaw, Paul C. 1998. Phonemic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexicon compression. *Computer Speech and Language*, 12:119–142.

Bowles, Richard L. and Robert I. Damper. 1989. Application of Dempster-Shafer theory of evidence to improved-accuracy, isolated-word recognition. In *Proceedings of Eurospeech '89*, Volume 2, pages 384–387, Paris.

Brill, Eric and Jun Wu. 1998. Classifier combination for improved lexical disambiguation. In *COLING-ACL '98: 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, pages 191–195, Montreal, Canada.

Byrd, Roy J. and Martin S. Chodorow. 1985. Using an on-line dictionary to find rhyming words and pronunciations for unknown words. In *Proceedings of the 23rd Meeting*, pages 277–283, Chicago, IL. Association for Computational Linguistics.

Carney, Edward. 1994. *A Survey of English Spelling*. Routledge, London, UK.

Chomsky, Noam and Morris Halle. 1968. *The Sound Pattern of English*. Harper and Row, New York.

Coltheart, Max. 1978. Lexical access in simple reading tasks. In G. Underwood, editor, *Strategies of Information Processing*. Academic Press, New York, pages 151–216.

Coltheart, Max. 1984. Writing systems and reading disorders. In L. Henderson, editor, *Orthographies and Reading*. Lawrence Erlbaum Associates, London, UK, pages 67–79.

Daelemans, Walter, Antal van den Bosch, and Ton Weijters. 1997. IGTree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.

Damper, Robert I. 1995. Self-learning and connectionist approaches to text-phoneme conversion. In Joe Levy, Dimitrios Bairaktaris, John Bullinaria, and Paul Cairns, editors, *Connectionist Models of Memory and Language*. UCL Press, London, UK, pages 117–144.

Damper, Robert I. and John F. G. Eastmond. 1996. Pronouncing text by analogy. In *Proceedings of the 16th International Conference on Computational Linguistics*, Volume 2, pages 268–273, Copenhagen, Denmark.

Damper, Robert I. and John F. G. Eastmond. 1997. Pronunciation by analogy: Impact of implementational choices on performance. *Language and Speech*, 40(1):1–23.

Damper, Robert I., Yannick Marchand, Martin J. Adamson, and Kjell Gustafson. 1999. Evaluating the pronunciation component of text-to-speech systems for English: A performance comparison of different approaches. *Computer Speech and Language*, 13(2):155–176.

Dedina, Michael J. and Howard C. Nusbaum. 1986. PRONOUNCE: A program for pronunciation by analogy. Speech Research Laboratory Progress Report No. 12, Indiana University, Bloomington, IN.

Dedina, Michael J. and Howard C. Nusbaum. 1991. PRONOUNCE: A program for pronunciation by analogy. *Computer Speech and Language*, 5:55–64.

Dietterich, Thomas G. 1997. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136.

Divay, Michel and Anthony J. Vitale. 1997. Algorithms for grapheme-phoneme translation for English and French: Applications for database searches and speech synthesis. *Computational Linguistics*, 23(4):495–523.

Dutoit, Thierry. 1997. *Introduction to Text-to-Speech Synthesis*. Kluwer, Dordrecht, The Netherlands.

Elovitz, Honey S., Rodney Johnson, Astrid McHugh, and John E. Shore. 1976. Letter-to-sound rules for automatic

translation of English text to phonetics. *IEEE Transactions on Acoustics, Speech and Signal Processing ASSP-24*, pages 446–459.

Glushko, Robert J. 1979. The organization and activation of orthographic knowledge in reading aloud. *Journal of Experimental Psychology: Human Perception and Performance*, 5:674–691.

Golding, Andrew R. 1991. *Pronouncing Names by a Combination of Case-Based and Rule-Based Reasoning*. Ph. D. thesis, Stanford University, CA.

Golding, Andrew R. and Paul S. Rosenbloom. 1991. Improving rule-based systems through case-based reasoning. In *Proceedings of the American Association for Artificial Intelligence Conference, AAAI-91*, pages 21–27, Anaheim, CA.

Hall, David L. 1992. *Mathematical Techniques in Multisensor Data Fusion*. Artech House, Boston, MA.

Hall, David L. and James Llinas. 1997. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23.

Hunnicutt, Sheri. 1976. Phonological rules for a text-to-speech system. *American Journal of Computational Linguistics Microfiche*, 57:1–72.

Jacobs, Robert A., Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87.

Jones, Daniel. 1996. *Analogical Natural Language Processing*. UCL Press, London, UK.

Kay, Janice and Anthony Marcel. 1981. One process, not two, in reading aloud: Lexical analogies do the work of non-lexical rules. *Quarterly Journal of Experimental Psychology*, 33A:397–413.

Kittler, Josef, Mohammad Hatef, Robert P. W. Duin, and Jiri Matas. 1998. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239.

Lehnert, Wendy G. 1987. Case-based problem solving with a large knowledge base of learned cases. In *Proceedings of the 6th National Conference on Artificial Intelligence, AAAI-87*, pages 301–306, Seattle, WA.

Liberman, Isabelle, Alvin M. Liberman, Ignatius Mattingly, and Donald Shankweiler. 1980. Orthography and the beginning reader. In James F. Kavanagh and Richard L. Venezky, editors, *Orthography, Reading and Dyslexia*. University Park Press, Baltimore, OH, pages 137–153.

Lucassen, John M. and Robert L. Mercer. 1984. An information theoretic approach to the automatic determination of phonemic baseforms. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing, ICASSP-84*, pages 42.5.1–42.5.4, San Diego, CA.

McCulloch, Neil, Mark Bedworth, and John Bridle. 1987. NETspeak—A re-implementation of NETtalk. *Computer Speech and Language*, 2:289–301.

McIlroy, M. Douglas. 1973. *Synthetic English Speech by Rule*. Computer Science Technical Report No. 14, Bell Telephone Laboratories.

Post, Emil. 1943. Formal reductions of the general combinatorial problem. *American Journal of Mathematics*, 65:197–268.

Romary, Laurent and Jean-Marie Pierrel. 1989. The use of the Dempster-Shafer rule in the lexical component of a man-machine oral dialogue system. *Speech Communication*, 8(2):159–176.

Sampson, Geoffrey. 1985. *Writing Systems*. Hutchinson, London, UK.

Scragg, David G. 1975. *A History of English Spelling*. Manchester University Press, Manchester, UK.

Sejnowski, Terrence J. and Charles R. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168.

Siegel, Sidney. 1956. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill Kogakusha, Tokyo, Japan.

Stanfill, Craig and David Waltz. 1986. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228.

Stanfill, Craig W. 1987. Memory-based reasoning applied to English pronunciation. In *Proceedings of the 6th National Conference on Artificial Intelligence, AAAI-87*, pages 577–581, Seattle, WA.

Stanfill, Craig W. 1988. Learning to read: A memory-based model. In *Proceedings of the Case-Based Reasoning Workshop*, pages 406–413, Clearwater Beach, FL.

Sullivan, Kirk P. H. and Robert I. Damper. 1993. Novel-word pronunciation: A cross-language study. *Speech Communication*, 13:441–452.

Turvey, Michael T., Laurie Beth Feldman, and Georgije Lukatela. 1984. The Serbo-Croatian orthography constrains the reader to a phonologically analytic strategy. In Leslie Henderson, editor, *Orthographies and Reading*. Lawrence Erlbaum Associates, London, UK, pages 81–89.

van den Bosch, Antal. 1997. *Learning to Pronounce Written Words: A Study in Inductive Language Learning*. Ph. D. thesis,

University of Maastricht, The Netherlands.

van den Bosch, Antal, Alain Content, Walter Daelemans, and Beatrice De Gelder. 1994. Measuring the complexity of writing systems. *Journal of Quantitative Linguistics*, 1(3):178–188.

van den Bosch, Antal and Walter Daelemans. 1993. Data-oriented methods for grapheme-to-phoneme conversion. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*, pages 45–53, Utrecht, The Netherlands.

van Halteren, Hans, Jakub Zavrel, and Walter Daelemans. 1998. Improving data driven wordclass tagging by system combination. In *COLING-ACL '98: 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, pages 491–497, Montreal, Canada.

Venezky, Richard L. 1965. *A Study of English Spelling-to-Sound Correspondences on Historical Principles*. Ann Arbor Press, Ann Arbor, MI.

Venezky, Richard L. 1970. *The Structure of English Orthography*. Mouton, The Hague, The Netherlands.

Wijk, Axel. 1966. *Rules of Pronunciation of the English Language*. Oxford University Press, Oxford, UK.

Yvon, François. 1996. Grapheme-to-phoneme conversion using multiple unbounded overlapping chunks. In *Proceedings of the Conference on New Methods in Natural Language Processing, NeMLaP '96*, pages 218–228, Ankara, Turkey.

Yvon, François. 1997. Paradigmatic cascades: A linguistically sound model of pronunciation by analogy. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 429–435, Madrid, Spain.