Computer Speech and Language (1999) **13**, 155–176
Article No. csla.1998.0117
Available online at http://www.idealibrary.com on IDEAL

Evaluating the pronunciation component of text-to-speech systems for English: a performance comparison of different approaches*

R. I. Damper,† ‡ ‖ Y. Marchand,†‖ M. J. Adamson†‖ and K. Gustafson§¶‖

†*Image, Speech and Intelligent Systems (ISIS) Research Group, Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.,* ‡*Center for Spoken Language Understanding, Oregon Graduate Institute of Science and Technology, PO Box 91000, Portland, OR 97291-1000, U.S.A.,* §*Department of Speech, Music and Hearing, KTH (Royal Institute of Technology), S-100 44 Stockholm, Sweden,* ¶*Telia Promotor AB, Box 2069, S-17102, Solna, Sweden*

Abstract

The automatic derivation of word pronunciations from input text is a central task for any text-to-speech system. For general English text at least, this is often thought to be a solved problem, with manually-derived linguistic rules assumed capable of handling "novel" words missing from the system dictionary. Data-driven methods, based on machine learning of the regularities implicit in a large pronouncing dictionary, have received considerable attention recently but are generally thought to perform less well. However, these tentative beliefs are at best uncertain without powerful methods for comparing text-to-phoneme subsystems. This paper contributes to the development of such methods by comparing the performance of four representative approaches to automatic phonemization on the same test dictionary. As well as rule-based approaches, three data-driven techniques are evaluated: pronunciation by analogy (PbA), NETspeak and IB1-IG (a modified k-nearest neighbour method). Issues involved in comparative evaluation are detailed and elucidated. The data-driven techniques outperform rules in accuracy of letter-to-phoneme translation by a very significant margin but require aligned text-phoneme training data and are slower. Best translation results are obtained with PbA at approximately 72% words correct on a resonably large pronouncing dictionary, compared with something like 26% words correct for the rules, indicating that automatic pronunciation of text is not a solved problem.

© 1999 Academic Press

*Based on a paper presented at UK Speech and Language Technology (SALT) Club Workshop on *Evaluation in Speech and Language Technology*, Sheffield, July 1997.

‖E-mails: {rid|ym|mja95r}@ecs.soton.ac.uk and kjellg@speech.kth.se

1. Introduction

The automatic conversion of text to a phonemic specification of pronunciation is a necessary step in all current approaches to text-to-speech (TTS) synthesis. For important languages like English and French, with only partial regularities between the spelling and sound systems, it is also a hard computational problem. Consequently, ever since the inception of TTS technology, text-to-phoneme conversion has attracted a great deal of attention: In his recent book, Dutoit (1997) emphasizes its importance when he writes (p. 13): “It is thus more suitable to define text-to-speech as the *production of speech by machines, by way of the automatic phonetization of the sentences to utter*.” Because of its long history, the intensity and diversity of research in this area, and the *ad hoc* nature of evaluations to date, there is a regrettable tendency among speech scientists and speech technologists to view the problem as essentially solved. Unfortunately, as will become clear in the course of this paper, this is far from the case.

For most words encountered in the input of a TTS system, a canonical pronunciation is easily obtained by dictionary look-up. (In this paper, we use the words *dictionary* and *lexicon* interchangeably.) If the word is absent from the system dictionary, but is a morphological derivative of a dictionary word, well-established techniques exist to infer a pronunciation (Allen *et al.*, 1987). Our concern here is the situation in which the input word is “novel”—such that dictionary look-up (possibly in conjunction with morphological analysis) fails to produce an output. In this commonly-encountered situation, a “back-up” (default) strategy must be employed.

The traditional back-up strategy, in practical use for many years, employs a set of context-dependent phonological rules written by an expert (e.g. Ainsworth, 1973; McIlroy, 1974; Elovitz *et al.*, 1976; Hunnicutt, 1976; Divay *et al.*, 1997). However, 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 and van den Bosch (1997) for more recent discussion. It is also conceivable that such data-driven techniques actually outperform traditional rules. However, this possibility does not seem to have been 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” (footnote 14, p. 115).

Whatever the veracity of these statements, there now exists a variety of methods of text-to-phoneme conversion to challenge the traditional rule-based methodology. System implementors are faced, therefore, with making a rational choice among these competing methods. This is problematic because techniques for evaluating the pronunciation component of a TTS system are very poorly developed: indeed, it is even unclear how different approaches with different motivations and characteristics should be compared. Such comparison is the focus of the remainder of this paper. Specifically, we present an empirical, quantitative evaluation of traditional rule-based methods and three newer data-driven methods: pronunciation by analogy (PbA), back-propagation neural networks (NETspeak) and a nearest-neighbour approach. Before proceeding, it is worth stating explicitly what we consider the pronunciation component of a TTS system is supposed to do, because this determines the way we evaluate it: it should deliver citation form pronunciations, that might plausibly have been produced by a lexicographer, for every word in the input.

2. Description of the techniques

In this section, we briefly describe the four automatic-phenomization techniques which are empirically compared later.

2.1. Phonological rules

The assumption here is that the pronunciation of a letter or letter substring can be found if sufficient is known of its context, i.e. the surrounding letters. 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. The form of the rules, strongly inspired by concepts from generative phonology (Chomsky & Halle, 1968, p. 14), is:

$$A[B]C \rightarrow D$$

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 strightforwardly cast in the IF. . . THEN form commonly featured in high-level programming languages and employed in expert, knowledge-based systems technology.

Because of the complexities of English spelling-to-sound correspondence (e.g. Carney, 1994), more than one rule generally applies at each stage of transcription. The potential conflicts which 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.* (1976) 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.

The rule set evaluated in this paper is that of Elovitz *et al.* (1976) obtained by anonymous `ftp` from directory `comp.speech/synthesis` at `svr-ftp.eng.cam.ac.uk`. There are 329 rules in this set.

2.2. Pronunciation by analogy

Pronunciation by analogy exploits the phonological knowledge implicity contained in a dictionary of words and their corresponding pronunciations. The underlying idea is that a pronunciation for an unknown word is assembled 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 concatenating the partial pronunciations. There are two basic versions of PbA: *explicit* and *implicit* (Damper & Eastmond, 1997, pp. 4–5). Explicit analogy (e.g. Dedina & Nusbaum, 1991) retains the lexicon in its entirety, typically in the form of a list of words and their spellings. PbA requires a dictionary in which text and phonemics have been aligned, so that pronunciations corresponding to matching orthographics substrings can be identified. However, the necessary computational steps to assemble a pronunciation can be anticipated and carried out in advance. Thus, in implicit analogy (e.g. Sullivan & Damper, 1993), the lexical database is precompiled to yield a generalized phonological knowledge base which is consulted during pronunciation generation. This done, the (explicit) dictionary can be discarded. In the terminology of machine learning, explicit and implicit approaches correspond to "lazy" and "eager" learning respectively (van den Bosch, 1997). Lazy learning minimizes the extent of any prior training phase, whereas eager learning involves significant prior training. Often, but not necessarily, eager learning will also attempt to compress the training data. That is, in the process of extracting regularities useful for generalization, a proportion of the training data will actually be discarded.

The variant of PbA evaluated here is based on PRONOUNCE (Dedina & Nusbaum 1991), but with several further enhancements as detailed by Marchand and Damper (submitted) and briefly outlined below. In PRONOUNCE, an incoming word is matched in turn against all orthographic entries in the lexicon which is an integral part of the subsystem (explicit analogy). 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 requires that text and phonemes are previously aligned in the dictionary. 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.

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 labelled 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 labelled with the phonemes intermediate between P_{im} and P_{jm} in the phoneme part of the matched substring. Note that arcs corresponding to bigrams are labelled with the empty string: the two symbols of the bigram label the nodes at either end. Additionally, arcs are labelled with a "frequency" count (Dedina and Nusbaum's term, and nothing to do with frequency of usage in written or spoken communication) which is incremented by one each time that substring (with that pronunciation) is matched during the pass through the lexicon.

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.

Because the minimum length of a matched substring in PRONOUNCE is two (corresponding to the two nodes at either end of an arc labelled with the empty string), there may not be a complete path through the lattice. Thus, we have a "silence" problem with this version of PbA. The problem is avoided in Sullivan and Damper's (1993) approach in which the lattice nodes represent the junctures (notional spaces) between words and arcs can therefore be labelled with single letter to single phoneme correspondences, so that (like letter-to-sound rules) there is always a default correspondence available. This, however, is at the cost of considerably larger pronunciation lattices.

The version of PbA evaluated here features several enhancements over PRONOUNCE, as described by Marchand and Damper (submitted). First, we use "full" pattern matching between input letter string and dictionary entries, as opposed to Dedina and Nusbaum's (1991) "partial" matching. That is, rather than starting with the two strings left-aligned, we start with the initial letter of the input string $\mathcal{I}$ aligned with the end letter of the dictionary entry $\mathcal{W}$. The matching process terminates not when the two strings are right-aligned, but when the end letter of $\mathcal{I}$ aligns with initial letter of $\mathcal{W}$. Thus, the number of right shifts is equal to the sum of the lengths of the two strings minus one. Second, multiple (five) heuristics are used to score the candidate pronunciations. Individual scores are then multiplied together to produce a final overall score. The best-scoring pronunciation on this basis is then selected as output. Marchand and Damper show that this "multi-strategy" approach gives statistically significant performance improvements over simpler versions of PbA.

Each word of the dictionary was processed in turn by removing it from the dictionary and assembling a pronunciation for it by analogy with the remainder. This can be seen as a version of the n-fold cross-validation technique described by Weiss and Kulikowski (1991), as used by van den Bosch (1997, p. 54) in the evaluation of text-to-phoneme subsystems, with $n = L$ where L is the size of the entire lexicon.

2.3. *Neural networks — NETspeak*

Sejnowski and Rosenberg (1987) described NETtalk, a feedforward neural network for text-to-phoneme conversion. Not only is this probably the best-known example of a data-driven text-to-phoneme converter, it is possibly the best-known application of neural nets also. In particular, Sejnowski and Rosenberg helped to popularize the error back-propagation training algorithm of Rumelhart, Hinton and Williams (1986) by demonstrating its success in learning to tackle the hard problem of automatic phonemization.

NETtalk's input consists of a window of an odd number of letters, where the central letter is the "target" and the letters to left and right provide the context. The input text is stepped through the window letter-by-letter. NETtalk was trained to associate the target letter with its correct pronunciation — here a single phoneme. Thus, like PbA, it requires its training data to have been previously aligned so that the letter and phoneme representations of each word have the same length. The standard NETtalk had a 7-letter window. However, smaller and larger windows were also studied (in the range 3 to 11 letters). Generally, performance was found to improve with window size.

In the usual case where the orthographic representation is longer than the phonemic one, *null* phonemes were added to make the two strings the same length. In the (rarer) case when the phoneme string is the longer of the two, Sejnowski and Rosenberg invented new "phonemes", e.g. they used /K/ for the $x \rightarrow$/ks/ sound in *sexual*. Each input character was represented by a sparse (1-out-of-n) code of 29 bits — one for each of the 26 letters plus three additional bits for punctuation marks and word boundaries. Thus, the number of input units i is $7 \times 29 = 203$. Sejnowski and Rosenberg used 21 "articulatory features" to represent the (single) phoneme output. Typical such features are voicing, place of articulation (labial, velar, ...) and tongue height. They also added five additional output units to represent stress and syllable boundaries. Hence, the number of output units o is $21 + 5 = 26$.

Although targets during training can (in principle) easily be specified by binary codings from the set of 2^{26} possible outputs, actual outputs are graded (because of NETtalk's sigmoidal activation functions) in the range (0, 1). Even if these were thresholded to give "hard" (0, 1) values, very few would correspond exactly to legal codings. So how do we know what the obtained output phoneme is? Sejnowski and Rosenberg's solution was to compute the inner product of the output vector with the codes for each of the possible phonemes. The phoneme that made the smallest angle with the output was chosen as the "best guess" output.

Various numbers of hidden layers and units (h) were studied. Specifically, single hidden layers with $h = 0$, 80 and 120 units were trained and tested, together with one net with two hidden layers each of $h = 80$ neurons. Performance increased with additional units and with an additional hidden layer. However, the most comprehensive results are presented for the case of a single hidden layer of 120 units.

With this latter net, the number of connections (excluding variable thresholds) is:

$$(i \times h) + (h \times o) = (203 \times 120) + (120 \times 26) = 27\,480.$$

To this must be added the small number ($h + o = 146$) of variable thresholds, one per unit. Thus, the number of adjustable weights (free parameters of the model) is nearly 28 000.

The neural network studied in the present work is based on NETspeak, McCulloch, Bedworth, and Bridle's (1987) re-implementation of NETtalk (but using a different dictionary and output phoneme set). McCulloch *et al.* (1987) additionally explored the impact of different input and output codings. Like NETtalk, NETspeak used a window of seven characters through which the input was stepped one character at a time. The input and output codings were thought to be important in that "an appropriate coding can greatly assist learning whilst an inappropriate one can prevent it". In place of NETtalk's 1-out-of-*n* coding leading to 203 input units, NETspeak used a more compact (2-out-of-*n*) 11-bit coding, giving $i = 77$. The first five bits indicated which of five rough, "phonological sets" the letter belonged to and the remaining six bits identified the particular character. In place of NET talk's 21 articulatory features to represent the (single) phoneme output (plus five stress and syllable boundary units), NETspeak used $o = 25$ output features. NETspeak used 77 hidden units in place of NETtalk's 120. Hence, NETspeak had just 7854 adjustable parameters compared with NETtalk's 27 480.

Learning parameters (learning rate, momentum, etc.) are all as in the original NETspeak. Our re-implementation differs from the original in two respects, however. First, instead of calculating the network error at each iteration of training and updating weights to all output nodes, only weights leading to individual nodes with erroneous outputs were updated. This was done after the presentation of each letter, as in NETspeak. Benefits of this training modification include quicker convergence and an improvement in final performance. Second, we introduced a termination condition based on network error, rather than using a fixed number (three) of iterations. The network output was evaluated on the test set after every five iterations of training. If the performance deteriorated on two consecutive occasions, the test result obtained 10 iterations previously (i.e. before performance started to deteriorate) was taken as the final result. This means that the net was trained on typically 40–50 (but as many as 700) iterations, as opposed to just three for NETspeak. Hence, we are confident that the net was reasonably well trained (even though we have not used an independent unseen set to monitor the training error).

2.4. Nearest neighbour — IB1-IG

In the IB1-IG approach (Daelemans *et al.*, 1997; van den Bosch, 1997), a feature-weighting function is used to provide a real-valued expression of the relative importance of feature values (i.e. letter positions) when performing the mapping from input to (phoneme) classification. Weighting is by information gain. The main idea is to interpret the training material

TABLE I. Instances generated in IB1-IG for the word *book*

Instance number	Left context	Focus letter	Right context	Phoneme class
1	- - -	*b*	*ook*	/b/
2	- -*b*	*o*	*ok*-	/u/
3	-*bo*	*o*	*k*- -	/-/
4	*boo*	*k*	- - -	/k/

(letter-phoneme correspondences) as an information source capable of generating a number of messages (i.e. phoneme classifications) with a certain probability.

Database information entropy is equal to the average number of bits of information needed to know the class given an instance. It is computed as:

$$E(\text{Database}) = -\sum_i p_i \log_2 p_i,$$

where the probability of phoneme class i, p_i, is estimated by its relative frequency in the training set.

For each feature, its relative importance in the database can be calculated by computing its information gain. To do this, we compute the average information entropy for this feature and subtract it from the information entropy of the database. The classification function of IB1-IG computes the similarity between a new instance and all stored instances, and returns the class label of the most similar instance. The instance database is generated by a fixed-size window capturing a focus letter surrounded by three left and three right neighbour letters. An example of instances generated for the word *book* is shown in Table I. The similarity function corresponds to the sum of the information gain values associated to the mismatched letter positions between new instances and stored instances.

We have slightly changed this method by creating 26 disjoint instance databases instead of just one. There is a one database for each focus letter. Thus, before using the similarity function, the database which corresponds to the focus letter of the new instance is selected, rather than considering all possible letters. This increases performance both in terms of words correct (marginally) and run time (considerably).

3. Adequacy of rules

Translation by rules manually written by an expert linguist has long been (and remains) the staple method of automatic phonemization in TTS synthesis. As such, rule-based approaches form the *de facto* (or "gold") standard against which other techniques should be measured. Given this, it is clearly important to know precisely how well rules perform. Yet Pols (1989) wrote (p. 60): "There is very little experience in evaluating the text-specific part (text preprocessing, grapheme-to-phoneme conversion) of a rule synthesizer". This situation has hardly changed since: for instance, van Bezooijen and van Heuven's recent (1998) contribution on "Assessment of Synthesis Systems" to Gibbon, Moore and Winski's extensive *Handbook of Standards and Resources for Spoken Language Systems* devotes less than two pages to grapheme-to-phoneme conversion (Section 5.5.1.2 of Volume III) and falls short of offering a strategy for evaluation. Thus, in Volume I of Gibbon *et al.*, Choukri (1998, p. 60) writes: "As ... detailed in Chapter III:5, there is no standard and agreed-upon methodology for the assessment of the linguistic module of a TTS *[system]*".

So how can we assess the performance of a rule-based pronunciation component of a TTS system, what might we expect it to be, and is typical performance adequate for high-quality TTS applications? A review of the literature reveals an enormous disparity in the claimed performance of knowledge-based pronunciation rules. For instance, Pollatsek and Rayner (1993, p. 415) write:

> ". . . the pronunciation of English is too variable to sensibly be explained by rules, with estimates ranging from 25% to 75% of the pronunciation of words being accounted for by rules. (The percentage varies of course with the complexity and number of rules hypothesized.)"

To give some idea of the range of opinions on this matter, Glushko (1981, p. 81) comments:

> ". . . rules alone simply don't work",

yet Klatt (1987, pp. 770–771) writes:

> ". . . the performance *[of NETtalk]* is not nearly as accurate as that of a good set of letter-to-sound rules (performing without use of an exceptions dictionary, but with rules for recognizing common affixes). A typical knowledge-based rule system (Bernstein *et al.*, 1980) is claimed to perform at about 85% correct at a word level . . . in a random sampling of a very large dictionary, which implies a phoneme correct rate of better than 97%."

It is difficult to resolve the obvious contradictions in these views. (Resolution is not helped by the fact that we are unable to locate in their paper the "85% correct at word level" claim that Klatt attributes to Bernstein and Pisoni. However, Bernstein has confirmed this figure in a personal communication.) Rule-sets like those of Bernstein and Pisoni often become embodied in TTS products and, for commercial reasons, are not available for testing by independent researchers. Hence, the above figures are not amenable to replication and verification.

This is not true of the Elovitz *et al.* (1976) rule set, which is freely available. Bernstein and Nessly (1981) wrote that the only published rule sets were those of McIlroy (1974), Hunnicutt (1976) and Elovitz *et al.* (1976). To this list should be added the even earlier Ainsworth (1973) rules for British English. To the best of our knowledge, the availability of rule sets remains today essentially as it was then.

Bernstein and Nessly (1981, Table II, p. 451) present evaluations done on various 1000-word subsets of the Brown corpus (Kučera *et al.*, 1967). Percentage correct word scores on each subset vary from 65% (for a sample of the rarest words) to 86·8% (for the 1000 most common words). Although the scoring was said to be (p. 449) "quite strict", it was actually rather lax as "the criterion for correctness was a good IPA transcription". (For instance, doubled consonants were not counted as errors.) In our view, this is unsatisfactory, as too much depends upon the arbitrary assignment "good". On the basis of frequency weighting, Elovitz *et al.* (1976) claim (p. 450): "we would expect to correctly translate 89 to 90 percent of the words in a random sample of English text". However, we feel that frequency weighting gives an unduly optimistic picture, as it is all too easy to ensure the correct pronunciation of very common words, which are not difficult to anticipate and which dominate the testing. Consider a trivially-simple pronunciation subsystem which has only a lexicon of the 10 most common words. It is guaranteed to make the pronunciation of these words, but no others, correct. It is easily estimated, either from Zipf's law (see Appendix A) or from data in the Brown corpus, that such an approach attains a word accuracy of approximately 20–30% on this basis alone when evaluated on general text. The problem for TTS systems is rare/unusual words which cannot be anticipated.

Hunnicutt (1980, pp. 52–55) presents a "preliminary evaluation" of her rules using subsets of 200 words from the Brown corpus. Percentage correct word scores varied from 66% (for the rarest words) to 100% (for the 200 most common words), but this was using a dictionary containing "the 200 words of highest frequency plus another 87 words which guarantee the correct pronunciation of the first 580 words according to frequency". Without this dictionary, using the rules alone, the figure for the 200 commonest words was 65·5%. Based on these results, Hunnicutt estimates the performance on words absent from the dictionary at 71% correct (p. 54). This estimate appears to be frequency-weighted.

There are obvious dangers in extrapolating from a small test set to general text (as we will demonstrate below). However, neither Elovitz *et al.* (1976) nor Hunnicutt (1980) report tests on the complete Brown corpus. To avoid the dangers inherent in extrapolation, this would seem to be well worthwhile, and vastly preferable to testing on subsets.

Bernstein and Nessly (1981) report a performance comparison of the Hunnicutt and Elovitz *et al.* rules. Outputs were assessed before stress assignment, and the assessment was (in our terms) lax as the intention was to evaluate the suitability of the "phenomicization. . . assuming one can assign stress correctly and then reduce vowels appropriately" (p. 20). Against this criterion, three experts rated the pronunciations as 1 = correct, 0·5 = close and 0 = wrong. Using this scoring scale, "both algorithms get about 25% words wrong on [*a*] lexically flat sample of 865 word types". Again, it is difficult to extrapolate this figure to performance on a large dictionary more representative of general text.

Finally, Divay and Vitale (1997, pp. 516–517) report an evaluation of their much newer rule set for English. They test on a 19 837 word subset of the Brown Corpus — presumably unique words, but the authors omit to say. Their pronunciation subsystem uses a dictionary, but its size is not explicitly specified. However, Divay and Vitale state that the "dictionary hit count was 7337 (36·99%)" which we can only interpret to mean that the dictionary contained 7337 words which were also in the Brown corpus. They further state that "the rules matched 5432 words (27·38%)". Hence, by simple arithmetic (and assuming that *matched* means *pronounced correctly*), they apparently matched 43·46% of the non-dictionary words — but these are presumably among the easier (more regular) words.

4. Comparing automatic-phonemization methods

Having considered how rules, the *de facto* standard method, might be assessed, we ask: what precisely are the difficulties in making comparisons between *different* approaches to the derivation of pronunciations from text? First, it is necessary to make a clear distinction between evaluating synthetic speech and evaluating a component of a TTS system. The former tests "fitness for purposes" of a complete system, whereas the latter is diagnostic and is our concern here. On this point, Bernstein and Nessly (1981, p. 19) write: "comparison of the output speech from two complete systems may not always provide a good test of the performance of the corresponding component algorithms in the two systems, because radical performance differences in other components can obscure small differences in the components of interest". This is not to say that the impact of the pronunciation component on overall quality and intelligibility of the speech output should not be assessed at some stage, but we believe this should be left to final system evaluation.

Turning to the pronunciation component itself, because the back-up strategy is only used when the primary approach of dictionary matching fails, it should (as Klatt implied) be assessed without the dictionary present. This poses problems for the rule-based approach. In practice, "exceptions" to the rules can either be embedded in the rule set (i.e. treated as highlyspecialized rules) or separated out and placed in the dictionary. For instance, Elovitz *et al.* include about 60 very common whole words as "rules"; Hunnicutt includes none. Divay and Vitale (1997, p. 505) write: "When we have a rule that handles *n* words, where *n* is between 1 and some small number, say fewer than 7, we generally put these words in a dictionary". As Klatt (1987, p. 772) states: "A moderate-sized exceptions dictionary can hide the deficiencies of a weak set of letter-to-sound rules, but at a high cost in terms of storage requirements". Hence, there is a real problem in deciding where the division between the two lies.

Moreover, Bernstein and Nessly (1981, p. 19) write: "Comparing the accuracy of different algorithms for text-to-phoneme conversion is often difficult because authors measure and report system performance in incommensurate ways". Further, Hunnicutt, Meng, Seneff & Zue (1993) state: "direct comparison of performance of different systems is difficult due to the lack of standardized phone sets, data sets or scoring algorithms". It is now established procedure to test the generalization ability of data-driven techniques with unseen words — not available during training. But how does the seen/unseen distinction apply in the case of a knowledge-based technique, if at all? How do we know what words (if any) and what accent of English the rule developer had in mind when writing the rules?

In spite of the difficulties, we nonetheless believe the need for comparative evaluation techniques is urgent. Given the above discussion, we propose:

(1) competitor techniques for automatic phonemization should be tested on the same (large) dictionaries in their entirety;
(2) scoring should be as strict as possible, should not be frequency weighted (for the reason outlined above), and should use a standardized metric;
(3) a standardized output (phoneme) set should be used.

Clearly, if any of three basic elements — dictionary, scoring metric or phoneme set — differs in the evaluation of different pronunciation subsystems, comparison is made difficult. As we will see, however, for practical reasons our evaluations actually fall a little short of the ideal expressed in (3).

Although we generally favour the Levenshtein (1966) string-edit distance, scoring in terms of words correct is easier, it avoids any necessity to enumerate phoneme insertions, deletions and/or substitutions, and is more stringent and sensitive than symbols correct scores (which are routinely in the greater than 90% range). This is the metric used here. A further advantage is that, as word pronunciations are either right or wrong, word accuracies are binomially distributed: this facilitates testing of statistical significance (see Appendix A). Finally, in addition to testing on the complete dictionary (as in (1) above), it is also of interest to know how performance for the data-driven methods varies as a function of the size of training data set and test set. In particular, this gives a way of estimating asymptotic performance as dictionary size tends to infinity.

5. Practical comparison

5.1. Materials

A machine-readable version of the (American English) *Teachers' Word Book* (TWB) of Thorndike and Lorge (1944) was used in this work. It had previously been manually aligned for the purpose of training NETspeak (McCulloch *et al.*, 1987). (Thanks to David Bounds for supplying us with this lexicon). Pronunciations were obtained for each of the 16 280 words using: the popular, publically-available rules of Elovitz *et al.*; pronunciation by analogy (PbA); NETspeak; and IB1-IG.

5.2. Harmonization of the phoneme set

For all four pronunciation subsystems, outputs were compared with their (known) dictionary pronunciations. At this stage, we have not included lexical stress markers in the set of output symbols: the important aspect of stress assignment is left for future work. The standardization of the phoneme set was not entirely without difficulty. The three data-driven methods necessarily produce output in terms of the TWB phoneme inventory of 52 phoneme symbols. The rules, however, produce output using an incommensurate alphabet of 41 phonemes. A further complication is the different transcription standards, and the different dialectal/accentual forms and distinctions, used by the dictionary- and rule-writers — so that the rule phoneme inventory is not just a proper subset of the dictionary inventory.

Proper evaluation requires that each subsystem is scored on the same output phoneme set, so that the TWB inventory and the Elovitz *et al.* inventory must somehow be made commensurate. We call this process *harmonization*. In this work, we do not achieve perfect harmonization. One approach would be to collapse the inventories into the smaller of the two sets so as to abolish any phonetic distinction not respected in the rule outputs. To some extent, this favours the rules over the data-driven methods, as (by their use of 52 output symbols rather than just 41) the latter are attempting to make finer distinctions but would not be penalized for getting them wrong. Also, the data-driven methods hold the considerable advantage over the rules that their pronunciations are assembled from the exact phoneme inventory embodied in the dictionary. Given these considerations, we decided it was fairest to evaluate the data-driven methods with the full inventory of 51 phonemes, but to evaluate the rules on a harmonized subset of these. Appendix B gives the phoneme sets and details of the harmonization process adopted.

5.3. Effect of dictionary size

Practical evaluation can only ever be on a dictionary of restricted, finite size, yet we really should characterize a pronunciation subsystem by its asymptotic performance on an effectively infinite-size test set. Hence, we should test on the largest lexicon available. Thus, there is a particular difficulty in comparing trainable data-driven methods with approaches that do not require training, like rules. The latter can be simply tested on the entire dictionary, but the former require some words to be held out as test data.

In principle, it is possible to use L-fold cross-validation as was done for PbA by Damper and Eastmond (1997). This means that training (lazy learning in this case) is always on the maximum number of $(L - 1)$ words, and all L words are tested as unseen. However, the eager (back-propagation) learning required by NETspeak is far too computationally intensive for this to be practical. The same is true of IB1-IG, where a set of $(L - 1)$ instance databases would have to be extracted from the dictionary at considerable computational cost. Further, it is of interest to know how performance depends on the size of test and/or training sets, because this gives some idea of asymptotic performance. In view of this, we have used the evaluation procedure described below, which was introduced by Lucas and Damper (1992).

By random sampling from the dictionary, two disjoint subsets of the same size N are produced. We train on one of these and test on the other. This is done for various values of N upto the maximum of half the size of the lexicon $L/2$. We then plot accuracy on the *train* and *test* subsets vs. N. The values of N used here were 100, 500, 1000, 2000, 5000 and 8140 words.

6. Results

In this section, we detail the results obtained with the four automatic phonemization approaches. Strenuous efforts were made to ensure that each data-driven technique had been properly implemented and was performing correctly. This took the form of, as far as possible, replicating (or improving) the results originally reported for each technique. These efforts are detailed under the relevant subheadings.

6.1. Phonological rules

The Elovitz *et al.* rules were evaluated on the complete dictionary of 16 280 words. Just 25·7% of the word pronunciations were correct. Length errors (especially due to geminate consonants), /g/$\leftrightarrow$/j/ confusions and vowel substitutions abound. This figure is so low that we were at considerable pains to confirm it. Two of the authors obtained the same result working independently to the agreed harmonization scheme in Appendix A. One of the authors gave his students (14 final year Computer Science with Artificial Intelligence undergraduates) the problem of evaluating the Elovitz *et al.* rules on TWB as an assessed programming exercise. Students had to devise their own harmonization methods and were not told what outcome to expect. All nine students who produced complete and correct code replicated the above figure to within 1.5 percentage points: the variation was attributable to differences in harmonization. Finally, Bagshaw (1998, p. 133) has recently confirmed the poor performance of the Elovitz *et al.* rules, obtaining just 19·34% words correct using “a slightly modified” version of this rule set on the CNET lexicon of 110 000 words. (He does not, however, detail his harmonization method.)

Arguably, one might expect publically-accessible rules, dating back essentially to 1976, to perform less well than rules included in a current TTS product. After all, the intervening 20 years have seen considerable research efforts directed at rule-based speech synthesis. For this reason, we also evaluated an up-to-date and commercially-successful rule-set. (It was a condition of access to these rules that they should remain anonymous.) However, these anonymous rules actually produced a lower word accuracy of 23·8% when evaluated using the above techniques. Specific details of the harmonization were, of course, rather different as the anonymous rules used a different output phoneme inventory (of size 49).

6.2. PbA

Alone among the data-driven techniques, the performance of PbA can be practically evaluated (i.e. in reasonable time) using L-fold validation on the complete dictionary of 16 280 words. This yielded a figure of 71·8% words correct. This result must be interpreted cautiously — the scoring of the rule outputs with reference to dictionary pronunciations using a different transcription standard is perhaps too stringent. However, the performance difference between 71·8% and 25·7% is so striking that it is almost superfluous to test its statistical significance. Using the test described in Appendix A, the probability of obtaining such a difference by chance is infinitesimally small ($z = 134{\cdot}6$).

As stated above, the version of PbA implemented here is that described by Marchand and Damper (submitted). This actually achieves the best performance so far reported for PbA on the TWB lexicon. Hence, we are confident that it has been properly implemented. By comparison, Damper and Eastmond (1997) obtained 67·9% words correct on TWB. Yvon (1996) achieved 64·0% words correct on the 20 000 word lexicon (Webster’s dictionary) used to train NETtalk

— using "multiple unbounded overlapping chunks" as the nodes of the pronunciation lattice. Damper and Eastmond's corresponding figure for this lexicon was 60·7%.

PbA was further tested on the different sized subsets of TWB in two ways:

(1) pronunciations were produced for each word of the *test* subsets by analogy with the (same-sized, disjoint) *train* subset;
(2) pronunciations were produced for each word of the *test* subset by analogy with the complete dictionary;

using L-fold cross-validation in both cases.

Figure 1(a) shows the results. The lower curve, obtained with method 1, illustrates that performance improves monotonically as the size of the lexical database (the *train* subset) increases. Most of the errors for the vary small *train/test* sets were silence. There seems to be no case for using anything other than the largest available dictionary. This finding is not entirely vacuous: the possibility exists with PbA that best results are obtained by analogy with a small lexicon of representative words and that the presence of other (exotic or rare) words is harmful. This does not appear to be so (although we need to be aware that here subsets are formed by random sampling rather than by principled selection). Method 2 is closer to the way one would use PbA in reality. The upper curve, obtained with method 2, is effectively constant apart from some sampling variance which is more pronounced for the smaller *test* subsets. We note that the two curves are asymptotic to the same value of about 72% words correct, lending credence to this value as a good estimate of ultimate performance on a very large dictionary.

6.3. NETspeak

To check the implementation, we first replicated the reported results of McCulloch *et al.* (1987). Figure 1 (b) shows the results subsequently obtained for the present evaluation. The lower curve is for testing on the unseen *(test)* subset and the upper curve is for testing on the training data. The curves are less smooth than Figure 1 (a) because of sensitivity to the initial random weights used in training. The best unseen performance is 46·0% words correct on the largest $(L/2)$ *test* subset: training and testing on the complete dictionary gave 54·4%. Hence, asymptotic generalization performance is expected to be about 50%, considerably poorer than PbA but very much better than rules.

6.4. IB1-IG

The result for IB1-IG trained and tested on disjoint, same-sized subsets of the dictionary is depicted in Figure 1(c) (bottom curve). The best performance is 57·4% words correct on the largest $(L/2)$ *test* subset, which is better than NETspeak but not as good as PbA. (According to the statistical test of Appendix A, the probability that this difference could be due to chance is practically indistinguishable from zero, $z = 16{\cdot}4$.) This ranking is consistent with van de Bosch's (1997) finding that any compression of the dictionary seems to be bad for performance, i.e. the two lazy learning techniques of PbA and IB1-IG outperform the eager (back-propagation) learning approach of NETspeak.

We also carried out a (slightly simplified) L-fold cross-validation test for the various sized test subsets, intended to parallel the method 2 testing for PbA as shown in the upper curve of Figure 1(a). This was done by first computing the instance database (including the entropy measures) for the entire dictionary. Each word of the subset was then tested against the entire database after removing all its instances from the database. The simplification was that

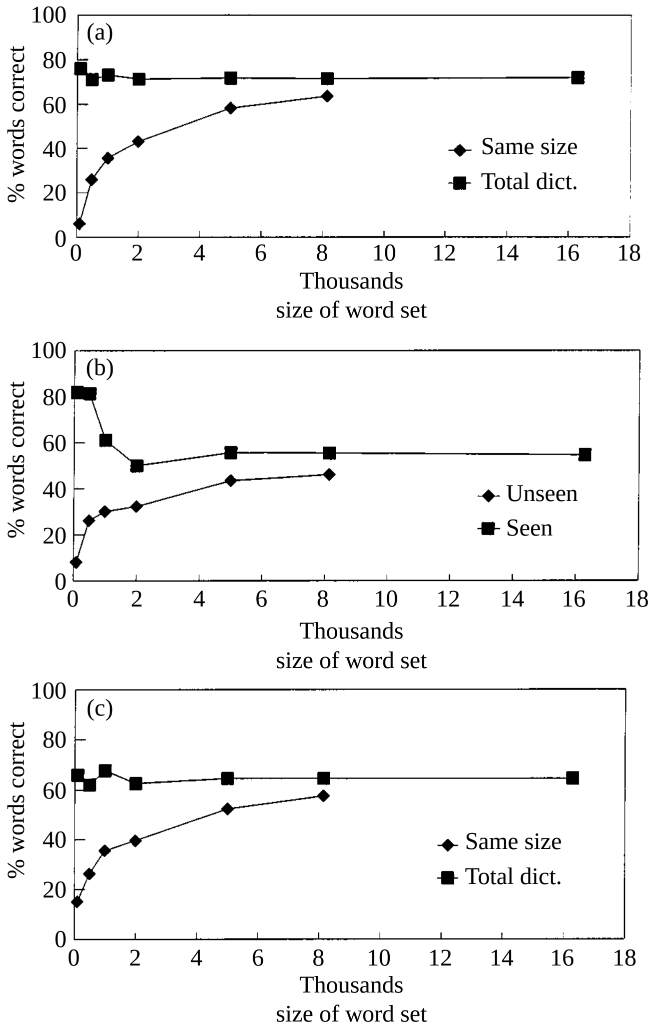

Figure 1. Results of training and testing on different-sized disjoint subsets of the dictionary: (a) PbA; (b) NETspeak; (c) IB1-IG. See text for specification of the testing strategies in each case.

entropies were not recomputed as each new test word was individually removed from the lexicon, because of the enormous computational cost of so doing. The assumption is that any single word will have little effect on the entropy calculation across 16 280 words. The result is depicted in Figure 1(c) (top curve). The asymptotic performance is about 65% words correct, which is better than NETtalk but poorer than PbA.

These results are poorer than those obtained by van den Bosch (1997, Fig. 3.3, p. 70) who obtained a higher word accuracy of 77·9% words correct with IB1-IG on CELEX (77 565 words). We attribute this apparently superior performance to use of a much smaller phoneme inventory of just 42 phonemes (including null) rather than to any problems with our implementation.

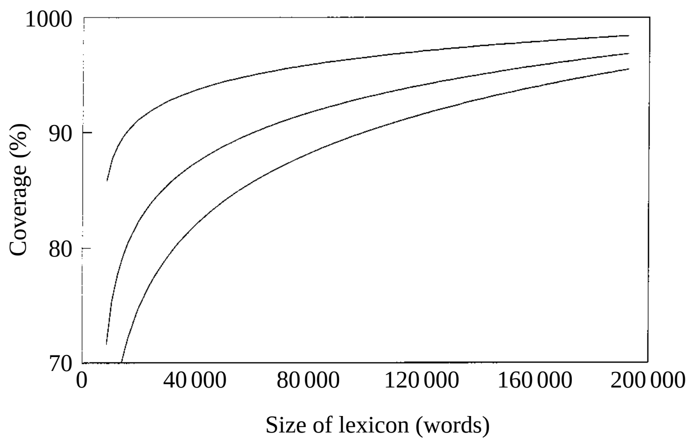

Figure 2. Theoretical coverage of dictionary plus back-up pronunciation strategy for unlisted words according to Zipf's law with $R = 200\,000$ words (see Appendix A for explanation). The upper curve assumes the word accuracy of the back-up strategy is 65%, the middle curve assumes it is 30% and for the lower curve it is 0% (i.e. the dictionary is used alone). This illustrates the importance of a good back-up strategy to performance of a TTS system.

6.5. Speed of conversion

Speed of conversion is an important aspect of the pronunciation component of a TTS system. It could be argued that PbA, for instance, is fundamentally slow and an advantage of rules is their computational efficiency. (For instance, Damper, Burnett, Gray, Straus & Symes (1987) implemented the Elovitz *et al.* rules in real time on a hand-held device within the constraints of mid-80s microprocessor technology, viz. 1 MHz processor with just 8 Kbytes of total memory.) In the present work, the total time to process the 16 280 words of TWB was 2·06 s on a Sun Ultra Enterprise with dual 170 MHz processor. (This machine is reserved for fast-response, light applications.) This corresponds to about 0·127 ms/word or a conversion speed of 7903 words/s, well in excess of normal speech rates and leaving plenty of processor time for other necessary tasks in a real-time TTS system. Nor should it be forgotten that the concern here is back-up strategies which are not continuously invoked.

NETtalk required 19·54 s to process the complete TWB on a 200 MHz Pentium, corresponding to 8·8 ms/word or a conversion speed of 112 words/s. This is slower that the rules but more than adequate for real-time TTS synthesis. Back-propagation training times were, however, very long: it took something like 9 days to produce the results of Figure 1 (b).

With PbA, it took 2490·8 s (42·5 min) to process the complete dictionary on a 75 MHz HP 712 workstation. This corresponds to 0·153 s/word or 6·54 words per second. With IB1-IG, it took 6626 · 0 s (110 · 4 min) to process the complete dictionary on the same HP workstation. This corresponds to 0·407 s/word or 2·56 words/s. However, the translation programs for PbA and IB1-IG were written for research purposes: they are implemented in the Python rapid prototyping language (rather than C) which is interpreted and relatively inefficient. The programs perform many data logging operations which would not be necessary in production versions of the code. Most of these features were disabled for the purposes of this testing, but not all. PbA in particular could be made very much faster by, for instance, precompling lexical knowledge (implicit analogy), using a fast string-searching algorithm like Boyer–Moore, etc.

7. Discussion and future work

In our view, there are two common misapprehensions in the speech synthesis research community. The first is that automatic phonemization is a solved problem; the second is that linguistic rules work much better than they actually do. For instance, on the first point, Liberman and Church (1991, p. 792) write:

> "We will describe algorithms for pronunciation of English words . . . that reduce the error rate to only a few tenths of a percent for ordinary text, about two orders of magnitude better than the word error rates of 15% or so that were common a decade ago."

Such error rates would, if they could be confirmed, render automatic phonemization an effectively solved problem.

In the event, however, Liberman and Church do not really describe their algorithms at all fully, in a way which would allow other investigators to reimplement them, and no formal evaluation supporting the claim of such a low error rate is detailed. They go on to describe the performance of NETtalk as (p. 819, citing a 1988 Technical Report from Princeton by Rosenberg) "so much worse . . ." than rules. This is similar to the opinion of Divay and Vitale (1997) quoted in the Introduction as well as to Klatt's (1987) earlier view quoted in Section 3, and to McCulloch *et al.*'s (1987, p. 301) statement: "Despite the remarkable results achieved by . . . NETtalk, *[they]* are still not as good as the best rule-based phonemic transcription systems".

We have shown in this paper that these opinions are mistaken. Further, a best word accuracy of just over 70% (the asymptotic performance of PbA) leaves much room for improvement, so that automatic pronunciation of text in TTS synthesis is not a solved problem. Much of the confusion on this point stems from failure to distinguish between the overall pronunciation component consisting of the dictionary plus back-up strategy and the back-up strategy itself. In spite of considerable development over many years, traditional rules (when properly assessed in the absence of the primary dictionary) perform very badly — much worse than pronunciation by analogy and other data-driven approaches like IB1-IG or even NETtalk — in spite of the many contrary views expressed in the literature.

It is worth remembering that the data-driven methods examined here require aligned (text-phoneme) data, whereas the rules do not. Thus, it could be argued that the data-driven methods are solving a simpler mapping problem, i.e. they have been unfairly "helped". Damper and Eastmond (1997, Table 2, p. 17) show that the use of unaligned training data, which then has to be aligned automatically, produces a reduction in performance from 67·9% words correct to 64·5%. This deficit could no doubt be reduced by improving the alignment algorithm.

Computer memory is becoming ever cheaper, and larger and better dictionaries are becoming available. Accordingly, one might argue that the importance of the back-up strategy is declining. Although this is undoubtedly true, it does not follow that performance of the back-up strategy is in any way unimportant. This is because, no matter how large the base dictionary of a TTS system is, its coverage can never be 100%. This issue is expanded upon in Appendix A. Figure 2 (generated as described in Appendix A) shows the theoretical coverage obtained as a function of dictionary size with the dictionary alone and the dictionary plus two different back-up strategies — one achieving 30% words correct (somewhat above our figures for rule-based subsystems) and the other achieving 65% words correct (a little below the performance of PbA). The positive impact that a good back-up strategy has is abundantly plain. Accordingly, it seems clear that data-driven methods such as PbA should replace rules in next-generation TTS systems.

The discrepancy between our measured figure of 25.7% words correct for the Elovitz *et al.* (1976) rules and that of 80–90% estimated by the original authors (as well as Bernstein and Nessly's (1981) value of 85%) is very considerable and merits some discussion. Although Elovitz *et al.* used frequency weighting, and this obviously played a part, Bernstein and Nessly did not. Hence, we are inclined to think that our stricter scoring is the principal reason for the discrepancy. Elovitz *et al.* were motivated by the desire to produce "good" pronunciations — on the grounds that these would be subjectively acceptable to listeners — rather than "perfect" ones, such as canonical dictionary entries. The same is true of Bernstein and Nessly's evaluation. Clearly then, future work will need also to assess the subjective acceptability of the pronunciations when used in synthesis, especially in view of Bernstein and Pisoni's statement (1980, p. 576) that: "Analysis of the results suggests that accurate acoustic realization of segmental information is the crucial factor in intelligibility". This is an area in which there has been significant and useful research activity (e.g. van Bezooijen & Pols, 1990; Pols, 1991; van Santen, 1993; Benoît, Grice & Hazan, 1996) to guide us. Indeed, subjective testing has often been the first-choice assessment strategy but, as argued earlier, we see its place as supporting the kind of evaluation reported here rather than replacing it.

There are many other priorities and avenues for future work, several of which are already underway. Although, for the reasons stated, we have a preference for words correct as an evaluation metric, it is nonetheless important to analyse the errors made by the pronunciation module at a symbol level. Clearly, it may be possible to eliminate whole categories of symbol error once these are identified. We are also now including stress in the conversion process and its evaluation. When this work started, the 16 280-word *Teachers' Word Book* was a relatively large dictionary. Now, however, something like 70–100 000 words is more typical of a TTS lexicon. Available dictionaries (see the on-line manual for the University of Edinburgh Festival TTS system at WorldWide Web URL `http://www.cstr.ed.ac.uk/projects/festival.html` for further documentation) include CUVOALD (70 000 words), CMU (∼ 100 000 words), and BEEP (163 000 words). Current work is using these much larger lexical resources.

Finally, we emphasize that this paper makes no consideration of the ultimate performance achievable by rules. The general rule formalism is very powerful: it can be Turing-equivalent. In fact, Post (1943) originally devised IF. . .THEN production rules specifically as a model of universal computation. However, in the form in which they are used in computational phonology, as described in Section 2.1, they are only regular (Johnson, 1972; Kaplan *et al.*, 1994; Luk *et al.*, 1996). Opinions vary on the computational power of feedforward networks like NETspeak, and this is probably not the place to develop the argument. The point is that manually-derived rules of the right specific form ought in theory to be capable of any computable string mapping and so should ultimately be as good as another approach. Our claim here is that *typical* linguists' rules as used in TTS systems — even those which have been the subject of intensive development efforts — are easily outperformed by current data-driven techniques.

This work was supported by research grant R000235487 "Speech Synthesis by Analogy" from the UK Economic and Social Research Council.

References

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

Allen, J., Hunnicutt, M. S. & Klatt, D. (1987). *From Text to Speech: The MITalk System*. Cambridge University Press, Cambridge.

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

Benoît, C., Grice, M. & Hazan, V. (1996). SUS test: a method for the assessment of text-to-speech synthesis using semantically unpredictable sentences. *Speech Communication* **18**, 381–392.

Bernstein, J. & Nessly, L. (1981). Performance comparison of component algorithms for the phonemicization of orthography. In *Proceedings of 19th Annual Meeting of the Association for Computational Linguistics*, Stanford, CA, pp. 19–21.

Bernstein, J. & Pisoni, D. B. (1980). Unlimited text-to-speech system: description and evaluation of a microprocessor-based device. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP-80*, Denver, CO, pp. 576–579.

Burgess, A. (1975). *Language Made Plain (Revised Edition)*. Fontana, London. (Pagination refers to 1984 Flamingo edition).

Carney, E. (1994). *A Survey of English Spelling*. Routledge, London.

Chomsky, N. & Halle, M. (1968). *The Sound Pattern of English*. Harper and Row, New York.

Choukri, K. (1998). System design. In *Spoken Language System Assessment*. (D. Gibbon, R. Moore and R. Winski, eds), pp. 30–78. Mouton de Gruyter, Berlin. Paperback Volume I of four volume *Handbook of Standards and Resources for Spoken Language Systems*.

Daelemans, W., van den Bosch, A. & Weijters, T. (1997). IGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review* **11**, 407–423.

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

Damper, R. I., Burnett, J. W., Gray, P. W., Straus, L. P. & Symes, R. A. (1987). A hand-held text-to-speech device for the non-vocal disabled. *Journal of Biomedical Engineering* **9**, 332–340.

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

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

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

Dutoit, T. (1997). *Introduction to Text-to-Speech Synthesis*. Kluwer, Dordrecht.

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

Glushko, R. J. (1981). Principles for pronouncing print: the psychology of phonography. In *Interactive Processes in Reading*. (A. M. Lesgold and C. A . Perfetti, eds), pp. 61–84. Lawrence Erlbaum, Hillsdale, NJ.

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

Hunnicutt, S. (1980). Grapheme-to-phoneme rules: a review. *Speech Transmission Laboratory Quarterly Progress and Status Report, Royal Institute of Technology (KTH), Stockholm STL-QPSR 2-3/1980*, pp. 38–60.

Hunnicutt, S., Meng, H., Seneff, S. & Zue, V. (1993). Reversible letter-to-sound sound-to-letter generation based on parsing word morphology. In *Proceedings of 3rd European Conference on Speech Communication and Technology (Eurospeech '93), Vol. 2*, pp. 763–766, Berlin, Germany.

Johnson, C. D. (1972). *Formal Aspects of Phonological Description*. Mouton, The Hague.

Kaplan, R. M. & Kay, M. (1994). Regular models of phonological rule systems. *Computational Linguistics* **20**, 331–378.

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

Kučera, H. & Francis, W. N. (1967). *Computational Analysis of Present-Day American English*. Brown University Press, Providence, RI.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory* **10**, 707–710.

Liberman, M. Y. & Church, K. W. (1991). Text analysis and word pronunciation in text-to-speech synthesis. In *Advances in Speech Signal Processing*. (S. Furui and M. M. Sondhi, eds), pp. 791–831. Marcel Dekker, New York.

Lucas, S. M. & Damper, R. I. (1992). Syntactic neural networks for bi-directional text-phonetics translation. In *Talking*

Machines: Theories, Models and Applications. (G. Bailly, C. Benoît and T. R. Sawallis, eds), pp. 127–141. Elsevier (North-Holland), Amsterdam.
Luk, R. W. P. & Damper, R. I. (1996). Stochastic phonographic transduction for English. *Computer Speech and Language* **10**, 133–153.
Marchand, Y. & Damper, R. I. (submitted). “A multi-strategy approach to improving pronunciation by analogy”. Manuscript submitted to *Computational Linguistics*.
McCulloch, N., Bedworth, M. & Bridle J. (1987). NETspeak — a re-implementation of NETtalk. *Computer Speech and Language* **2**, 289–301.
McIlroy, M. (1974). *Synthetic English Speech by Rule*. Bell Telephone Laboratories Memo.
Partee, B. H., ter Meulen, A. & Wall, R. (1993). *Mathematical Methods in Linguistics*. Kluwer, Dordrecht. (Corrected second printing).
Pollatsek, A. & Rayner, K. (1993). Reading. In *Foundations of Cognitive Science (2nd Edition)*. (M. I. Posner, ed.), pp. 401–436. Bradford Books/MIT Press, Cambridge, MA.
Pols, L. C. W. (1989). Assessment of text-to-speech synthesis systems. In *Speech Input and Output Assessment: Multilingual Methods and Standards*. (A. J. Fourcin, G. Harland, W. Barry and V. Hazan, eds), pp. 53–81. Ellis Horwood, Chichester.
Pols, L. C. W. (1991). Quality assessment of text-to-speech synthesis by rule. In *Advances in Speech Signal Processing*. (S. Furui and M. M. Sondhi, eds), pp. 387–416. Marcel Dekker, New York.
Post, E. (1943). Formal reductions of the general combinatorial problem. *American Journal of Mathematics* **65**, 197–268.
Rumelhart, D. E., Hinton, G. E. & Williams, R. (1986). Learning representations by back-propagating errors. *Nature* **323**, 533–536.
Schroeder, M. (1991). *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*. W. H. Freeman, New York.
Sejnowski, T. J. & Rosenberg, C. R. (1987). Parallel networks that learn to pronounce English text. *Complex Systems* **1**, 145–168.
Siegel, S. (1956). *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill Kogakusha, Tokyo.
Sullivan, K. P. H. & Damper, R. I. (1993). Novel-word pronunciation: a cross-language study. *Speech Communication* **13**, 441–452.
Thorndike, E. & Lorge, I. (1944). *The Teachers’ Word Book of 30,000 Words*. Teachers’ College, Columbia University, NY.
van Bezooijen, R. & Pols, L. C. W. (1990). Evaluating text-to-speech systems: some methodological aspects. *Speech Communication* **9**, 263–270.
van Bezooijen, R. & van Hueven, V. (1998). Assessment of synthesis systems. In *Spoken Language System Assessment*, (D. Gibbon, R. Moore and R. Winski, eds), pp. 167–249. Mouton de Gruyter, Berlin. Paperback Volume III of four volume *Handbook of Standards and Resources for Spoken Language Systems*.
van den Bosch, A. (1997). *Learning to Pronounce Written Words: A Study in Inductive Language Learning*. PhD thesis, University of Maastricht, The Netherlands.
van Santen, J. P. H. (1993). Perceptual experiments for diagnostic testing of text-to-speech systems. *Computer Speech and Language* **7**, 49–100.
Weiss, S. & Kulikowski, C. (1991). *Computer Systems that Learn*. Morgan Kaufmann, San Mateo, CA.
Yvon, F. (1996). Grapheme-to-phoneme conversion using multiple unbounded overlapping chunks. In *Proceedings of Conference on New Methods in Natural Language Processing (NeMLaP ’96)*, Ankara, Turkey, pp. 218–228.
Zipf, G. K. (1949). *Human Behaviour and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA.

(Received 12 November 1998 and accepted for publication 23 November 1998)

Appendix A. Impact of back-up strategy

According to Zipf’s law (Zipf, 1949; Schroeder, 1991, p. 35), for English:

$$f(r) \sim \frac{1}{r \log_e(1{\cdot}78R)},$$

where $f(r)$ is the frequency with which a word of rank r appears in a text composed of words selected from a set (“vocabulary”) of size R.

We note that R is something of an abstraction since, for a natural language like English, the vocabulary size is denumerably (or countably) infinite (Partee *et al.*, 1993, p. 59). That is, the lexical entries can be put in one-to-one correspondence with the natural numbers. A property of the natural numbers is that, given any finite list in ascending order $(1, 2, 3, \ldots)$, we can always generate a new number which is the successor of (i.e. one greater than) the last entry. As a consequence (and simplifying somewhat by ignoring issues of lexicographic order, phonotactic legality etc.), given any lexicon no matter how large, we can always generate a new word that is not listed. To quote the accomplished novelist Burgess (1975, pp. 26–27) on this point:

> "A language consists of potentialities . . . we cannot be sure of its content. How many words are there in English? We cannot say. It is not enough to point to the number of words of a dictionary, because no dictionary — however large — can pretend to be complete . . . in my own home, my family uses invented words like *shlerp*, *focklepoff* and *arpworthy* . . . just as *chortle*, *brilling* and *abnihilise* belong to English."

Suppose we have a lexicon of $L(\leq R)$ words. The coverage due to the lexicon is then:

$$\begin{aligned} C_L &\sim \int_1^L \frac{dr}{r \log_e(1{\cdot}78\ R)} \times 100\% \\ &\sim \frac{100 \log_e L}{\log_e(1{\cdot}78\ R)}. \end{aligned}$$

Note

(1) As rank is a discrete variable, the integration is here an approximation to discrete summation over the words of the lexicon. The approximation improves as L increases.
(2) C_L only approaches 100% for $L = R$, but even then never reaches it, reflecting the scale-invariant nature of the Zipf power law. (In other words, reflecting the fact that the set of words of a natural language is a countable infinity and cannot be listed.)
(3) For a lexicon of the size typically found in up-to-date TTS systems (e. g. CUVOALD, $L \sim 70\,000$ words), C_L is estimated at 87·3%, leaving almost 13% of the input to be pronounced by some means other than dictionary look-up.

Now, if the remaining $(1 - C_L)$ words are pronounced by the back-up strategy of letter-to-sound rules, neural networks, analogy etc., and this back-up strategy has accuracy A, then the total coverage is:

$$C_A = C_L + A(1 - C_L).$$

Figure 2 shows how the total coverage C_A varies with the size of lexicon L, assuming $R = 200\,000$ words, for the three cases of $A = 0{\cdot}65$, $A = 0{\cdot}3$ and $A = 0$ (i.e. dictionary used alone). For the typical size of lexicon considered above $(L \sim 70\,000)$:

$$\begin{aligned} C_0 &= 87{\cdot}28\% \\ C_{0{\cdot}3} &= 91{\cdot}09\% \end{aligned}$$

and

$$C_{0{\cdot}65} = 95{\cdot}55\%.$$

As only two outcomes are possible for the translation of each word — either the pronunciation is correct or it is not — the sampling distribution of the word accuracies is binomial. Hence, we can use a binomial test (Siegel, 1956, pp. 36–42) to determine the significance of the above differences. Because the number of trials (i.e. word translations) is large, we can use the normal

TABLE II. Elovitz phoneme symbols and TWB equivalents

Elovitz symbol	As in . . .	TWB symbol	TWB symbol in Elovitz set?
AA	father	AR	N
AE	fat	AA	Y: rewrite AA $\rightarrow$ AE before AR $\rightarrow$ AA
AH	but	_U	N
AO	lawn	AW	Y: rewrite AW $\rightarrow$ AO before OU $\rightarrow$ AW
AW	how	OU	N
AX	about	_A	N
AY	hide	IE	N
CH	char	CH	identical
DH	either	DH	identical
EH	get	_E	N
ER	murder	ER	N
EY	gate	Al	N
HH	how	_H	N
IH	hit	_I	N
IY	beet	EE	N
NG	sung	NG	identical
OW	lone	OA	N
OY	toy	OI	N
SH	leash	SH	identical
TH	ether	TH	identical
UH	full	OO	N
UW	fool	UU	N
WH	where	_W	(also maps to w)
ZH	leisure	ZH	identical
b	back	_B	N
d	dime	_D	N
f	fault	_F	N
g	goat	_G	N
j	jar	_J	N
k	coat	_K	N
l	laugh	_L	N
m	more	_M	N
n	sun	_N	N
p	pack	_P	N
r	rate	_R	N
s	sue	_S	N
t	time	_T	N
v	vault	_V	N
w	wear	_W	(also maps to WH)
y	young	_Y	N
z	zoo	_Z	N

approximation to the binomial distribution. The appropriate statistic is (Siegel, 1956, p. 41):

$$z = \frac{(x \pm 0{\cdot}5) - NP}{\sqrt{NPQ}}.$$

Considering the difference between $C_{0\cdot3}$ and $C_{0\cdot65}$, we use $N = 70\,000$ words, $P = 0{\cdot}9109$, $Q = (1 - P) = 0{\cdot}0891$ and $x = N \times 0{\cdot}9555$, to obtain $z = 41{\cdot}4$. The (one-tailed) probability that this enormously high value could have been obtained by chance is indistinguishable from zero; it is too small to be computed in Matlab. (Note that all computation in Matlab is double-precision). In fact, the critical value for the $P = 0{\cdot}0001$ significance level is $z = 3{\cdot}72$. Hence,

TABLE III. Phoneme symbols used by TWB but not by Elovitz

TWB symbol	As in . . .	Best Elo. equivalent
_O	falter	AO
EI	fairy	EHr
EY	despite	EH
IA	here	IYr
KH	anxious	kSH
KS	experiment	ks
KW	quench	kw
UL	shuttle	AXl
UR	assure	ER
YU	ejaculation	yUW
GZ	exalt	gz

an improved back-up strategy is capable of making a very highly significant contribution to pronunciation performance.

Appendix B. Harmonization details

Note

(1) In principle, the harmonization process consists of mapping the canonical pronunciations from TWB, which use a phoneme inventory of 52 symbols, into a set of pronunciations using the 41 phoneme symbol set of Elovitz *et al.* so that the rule outputs can be scored against the (harmonized) dictionary pronunciations. (But see (4) below.)

(2) The codes are not auto-segmental. For instance, the TWB substring ..OOUU.. could be OO+UU or part of AO+OU+UH. This is not a real problem, however, because all phoneme codes are of length two. Segmentation was therefore handled in harmonization to a common phoneme set by initially inserting phoneme boundary markers (+), at every second position, which were removed at the end of the process.

(3) In the case where a TWB symbol is not in the Elovitz phoneme set, the order of conversion is important. In UNIX terminology, it is possible for a TWB symbol rewrite to *clobber* another symbol, making it indistinguishable between the two symbol sets. The last column of the first table details these cases and specifies the order of rewrites used to avoid the clobber problem.

(4) The mapping from _W in the TWB alphabet to the Elovitz alphabet is not one-to-one. It rewrites both to WH and to w. This was handled by reducing the Elovitz alphabet to 40 phonemes (substituting w for WH throughout).