A COMPARISON OF LETTER-TO-SOUND CONVERSION TECHNIQUES FOR ENGLISH TEXT-TO-SPEECH SYNTHESIS

R.I. Damper †, Y. Marchand †, M.J. Adamson †, K. Gustafson *

†Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK
*Department of Speech, Music and Hearing, KTH, S-100 44 Stockholm, Sweden

1. INTRODUCTION

The automatic conversion of text to a phonemic specification of pronunciation is a necessary early 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, text-to-phoneme conversion has attracted a great deal of attention. Because of its long history, there is a regrettable tendency to view the problem as essentially solved. Unfortunately, this is far from the case.

For most words encountered in the input of a TTS system, a canonical (or 'baseform') pronunciation is easily obtained by dictionary look-up. 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 default or 'back-up' strategy must be employed. The traditional default strategy, uses a set of context-dependent phonological rules written by an expert (e.g. [1, 2, 3]). 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.

Hence, 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 poorly developed. To address this problem, we present here a quantitative evaluation of traditional rule-based methods and three newer data-driven methods: pronunciation by analogy, neural networks (NETspeak) and a nearest-neighbour approach.

2. DESCRIPTION OF THE TECHNIQUES

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. The form of the rules 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.

Because of the complexities of English spelling-to-sound correspondence, more than one rule generally applies at each stage of transcription. The 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. Transcription is most usually a one-pass, left-to-right process. For the particular target letter, the appropriate sublist is searched from top-to-bottom until a match is found. This rule is then fired, 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 rules evaluated in this paper are those of Elovitz *et al.* [2] obtained by anonymous `ftp` from directory `comp.speech/synthesis` at `svr-ftp.eng.cam.ac.uk`.

2.2 Pronunciation by Analogy

Pronunciation by analogy (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 assembled by matching substrings of the input to substrings of known, lexical words, hypothesising a partial pronunciation for each matched substring from the phonological knowledge, and concatenating the partial pronunciations.

The variant of PbA evaluated here is based on Dedina and Nusbaum's PRONOUNCE [4], but with several further enhancements as detailed by Marchand and Damper [5]. In PRONOUNCE, a data structure called the pronunciation lattice is built from matching substrings in the input word and the dictionary entries. This is a graph containing information about the position and total number of matched substrings, and their partial pronunciations. 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.

The version of PbA evaluated here features several enhancements over PRONOUNCE. First, we use 'full' pattern matching between input letter string and dictionary entries, as opposed to Dedina and Nusbaum's 'partial' matching. 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 [5] 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 of Weiss and Kulikowski [6]), but with $n = L$ where L is the size of the entire lexicon.

2.3 Neural Networks – NETspeak

The neural network studied in the present work is based on NETspeak, McCulloch *et al.*'s re-implementation of Sejnowski and Rosenberg's well-known NETtalk [7]. McCulloch *et al.* additionally explored the impact of different input and output codings (and examined the relative performance of separate networks for the transcription of common and uncommon words respectively). Like NETtalk, NETspeak used a window of 7 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 77 input units. 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 NETtalk's 21 "articulatory features" to represent the (single) phoneme output (plus five stress and syllable boundary units), NETspeak used 25 output features. NETspeak used 77 hidden units in place of NETtalk's 120. Hence, NETspeak had just 7854 adjustable parameters compared to 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, a different termination condition was used. 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 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.

2.4 Nearest Neighbour – IB1-IG

In the IB1-IG approach [8], 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 classification. Weighting is by information gain. The main idea is to interpret the training material (letter-phoneme correspondences) as an information source capable of generating a number of messages (i.e. 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. 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 base is generated by a fixed-size window capturing a focus letter surrounded by three left and three right neighbour letters. The similarity function corresponds to the sum of the information gain values associated to the mismatched letter positions between new instance and stored instances. We have slightly changed this method by creating 26 disjoint databases instead of one: one 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 26 letters. This increases performances both in term of words correct (marginally) and run time (considerably).

3. COMPARING AUTOMATIC-PHONEMISATION METHODS

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 purpose' of a complete system, whereas the latter is diagnostic and is our concern here. On this point, Bernstein and Nessly [9] 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 default strategy is only used when the primary approach of dictionary matching fails, it should 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 highly-specialised rules) or separated out and placed in the dictionary. For instance, Elovitz *et al.* include about 60 very common whole words as 'rules'. As Klatt [10] 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 [9] 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", while Hunnicutt *et al.* [11] 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 generalisation ability of data-driven techniques with unseen words – not available to the system 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) 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 that ideally:

- competitor techniques for automatic phonemisation should be tested on the same (large) dictionaries in their entirety;
- scoring should be as strict as possible, should not be frequency weighted (for the reason outlined above), and should use a standardised metric;
- a standardised 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 becomes difficult. In this work, we are not able to achieve a fully standardised phoneme set. As we shall see, for unavoidable reasons, the rules are evaluated on a slightly smaller set.

Scoring in terms of words correct is simple, 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 percent range). This is the metric used here. Finally, and notwithstanding the desirability of measuring performance on a large dictionary, it is also of interest to know how performance varies as a function of the size of training data set (if training is required) and test set. In particular, this gives a way of estimating asymptotic performance as dictionary size tends to infinity.

4. PRACTICAL COMPARISON

4.1 Materials

A machine-readable version of the (American English) *Teachers' Word Book* (TWB) [12] was used in this work. It had previously been manually aligned for the purpose of training NETspeak. Pronunciations were obtained for each of the 16,280 words using the four methods detailed above.

4.2 Harmonisation of the phoneme set

For all four pronunciation subsystems, outputs were compared with their (known) dictionary pronunciations. 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 standardisation 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 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 *harmonisation*. In this work, we do not achieve perfect harmonisation.

One approach would be to collapse the inventories into the smaller of the two sets so as to abolish any phonemic distinction not respected in the rule outputs. To some extent, this favours the rules over the data-driven methods, since (by their use of 52 output symbols rather than just 41) the latter are attempting to make finer distinctions but would not be penalised 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 to evaluate the data-driven methods with the full inventory of 51 phonemes, but to evaluate the rules on a harmonised subset of these. Details of the harmonisation procedure are given by Damper and Gustafson [13].

4.3 Effect of Dictionary Size

Practical evaluation can only ever be on a dictionary of restricted, finite size, yet we really should characterise 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 don't 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 [14]. This means that analogy is always on the basis of the maximum number of $(L-1)$ words, and all L words are tested as unseen. However, the back-propagation learning required by NETspeak is far too computationally intensive for this to be practical. The same is true of IB1-IG, where instances have to 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, since 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 [15].

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 up to the maximum of half the size of the lexicon $L/2$. We then plot accuracy on the *test* subset (and, if sensible, the *train* subset also) versus N. The values of N used here were 100, 500, 1000, 2000, 5000 and 8,140 words.

5. RESULTS

5.1 Context-Dependent 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 strenuous efforts were made to confirm it. Two of the authors obtained the same result working independently. One of the authors gave his students the problem of evaluating the Elovitz *et al.* rules on TWB as an assessed programming exercise. Those who produced correct code replicated the above figure to within 1 or 1.5 percentage points: differences were attributable to different specific harmonisation schemes. Finally, Bagshaw [16] 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.

5.2 PbA

Alone among the data-driven techniques, the performance of PbA can be practically evaluated using L-fold validation on the complete dictionary of 16,280 words. This yielded a figure of 71.8% words correct. While this result must be treated with a little caution – the scoring of the rule outputs with reference to dictionary pronunciations using a different transcription standard is probably too stringent – the difference from 25.7% is striking.

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 N of the lexical database (the *train* subset) increases, reaching 63.6% when $N = L/2$.

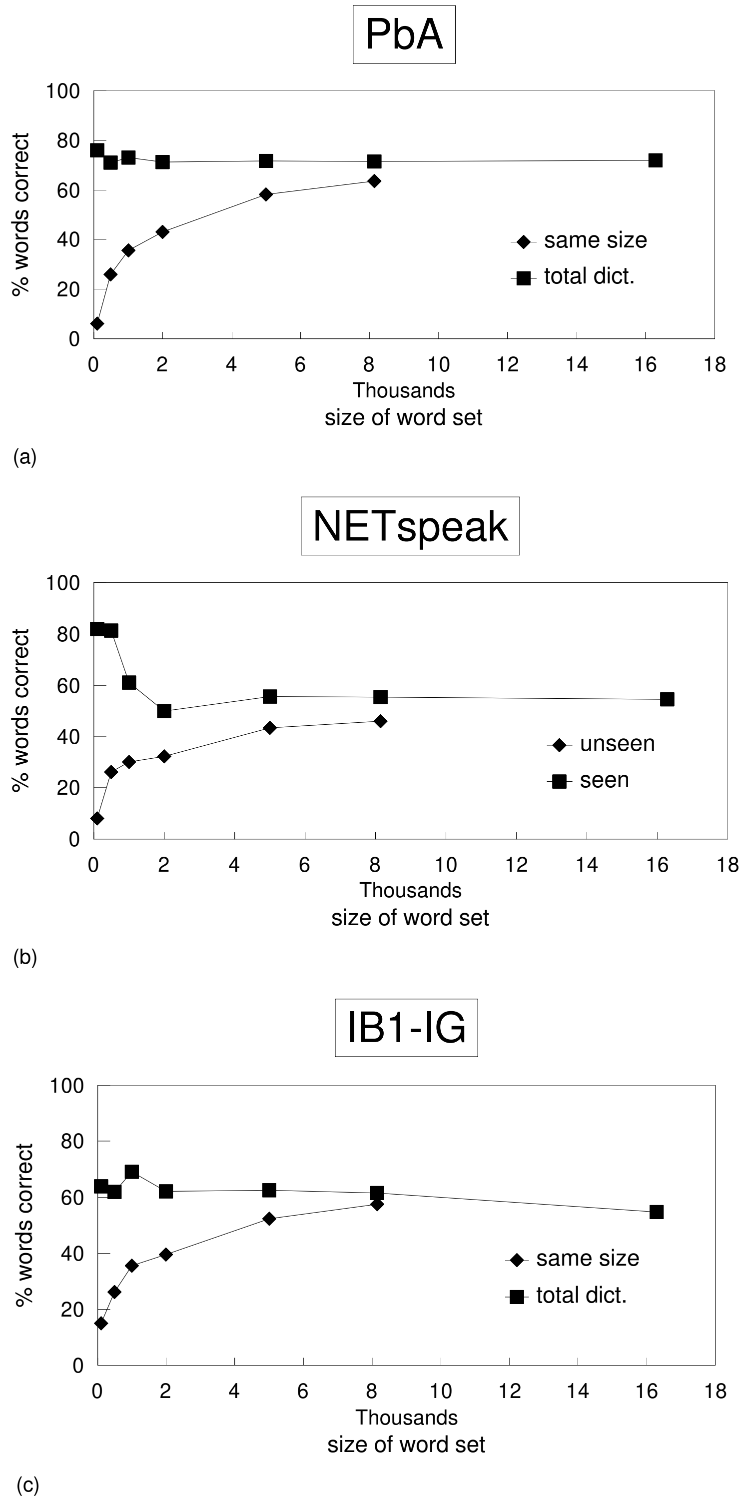

Figure 1: Results of training and testing on different-sized disjoint subsets of the dictionary: (a) PbA (b) NETspeak (c) IB1-IG. See text for specification of the testing strategies in each case.

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. 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.

5.3 NETspeak

Figure 1(b) shows the results. 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 46.03% words correct on the largest ($L/2$) *test* subset, while training and testing on the complete dictionary gave 54.44%. Hence, asymptotic generalisation performance is expected to be about 50%, considerably poorer than PbA.

5.4 IB1-IG

Here it was deemed sensible to test on the unseen data only because of the computational cost of pre-compiling the instances database for L-fold cross validation. The result is depicted in Figure 1(c). 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.

6. DISCUSSION AND FUTURE WORK

In our view, there are two common misapprehensions in the speech synthesis research community. The first is that automatic phonemisation 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 [17] 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 phonemisation an effectively solved problem.

In the event, however, Liberman and Church do not really describe their algorithms at all fully, and certainly not in a way which would allow other investigators to reimplement them. No formal evaluation supporting the claim of such a low error rate is detailed. One can only assume that their figure of "a few tenths of a percent" is impressionistic and, since they are at pains to say their methods are entirely 'dictionary-based', they must surely be assuming the presence of the system dictionary. They go on to describe the performance of NETtalk as (citing a 1988 Technical Report from Princeton by Rosenberg) "so much worse ..." than rules. This is similar to the opinion of Divay and Vitale [3] quoted in the Introduction and to McCulloch *et al.*'s [18] statement: "Despite the remarkable results achieved by ... NETtalk, *[they]* are still not as good as the best rule-based phonemic transcription systems".

In this paper, we have shown that automatic pronunciation of novel words is not a solved problem in TTS synthesis. The best that can be done is about 70% words correct using PbA. 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 system dictionary) perform very badly – much worse than pronunciation by analogy and other data-driven approaches like IB1-IG or even NETspeak (in spite of the many contrary views expressed in the literature).

With time, computer memory is becoming ever cheaper, and larger and better dictionaries are becoming available. Accordingly, it could be argued that the importance of the back-up strategy is declining. Although

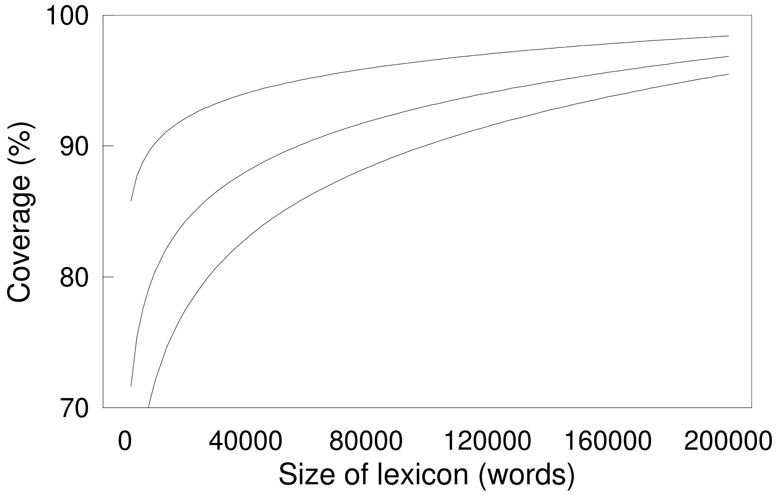

Figure 2: Theoretical coverage of dictionary plus back-up pronunciation strategy for unlisted words according to Zipf's law assuming 200 000 words of English. The upper curve assumes the word accuracy of the back-up strategy is 65% while the lower curve assumes the accuracy of the back-up strategy is 30%. This illustrates the importance of a good back-up strategy to performance of a TTS system

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%. Figure 2 (generated with a simple model based on Zipf's law) shows the theoretical coverage obtained as a function of dictionary size with two back-up strategies – one achieving 30% phonemes correct (somewhat above our figures for rule-based systems) and the other achieving 65% phonemes correct (below what could be expected 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 Elovitz *et al.* rule set performed very badly. Our evaluation tends to confirm the suspicion that existing rule-based TTS systems only do as well as they do because of the dictionary. The discrepancies between our measured figures for the Elovitz *et al.* rules and those estimated by the original authors (approximately 90% words correct) are very considerable and merit some discussion. While Elovitz *et al.* used frequency weighting, and this obviously played a part, 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. This is an area in which there has been significant and useful research activity (e.g. [19, 20, 21, 22]) 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 16 000-words or so *Teachers' Word Book* was a relatively large dictionary. Now, however, 70–100 000 words is more typical of a TTS lexicon. Available dictionaries now include CUVOALD (70 000 words), CMU (∼ 100 000 words), and BEEP (163 000 words). Current work is using these much larger lexical resources. We also intend including stochastic transduction – another very promising data-driven technique – in future evaluative comparisons.

Finally, we emphasise that this paper makes no consideration of the asymptotic performance achievable by either rules or any competitor technique. That is, it is entirely possible that manually-derived linguistic rules, as a result of their specific formalism, are inherently capable of asymptotic performance which outstrips that of other string-mapping devices. While this is unlikely, our claim here is that *typical* rules – even those which have been the subject of intensive development efforts – are easily outperformed by the best current data-driven techniques.

ACKNOWLEDGEMENTS

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

REFERENCES

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

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

[3] M. Divay and A. J. Vitale. Algorithms for grapheme-phoneme translation for English and French: Applications for database searches and speech synthesis. *Computational Linguistics*, 23:495–523, 1997.

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

[5] Y. Marchand and R. I. Damper. "A multi-strategy approach to improving pronunciation by analogy". Manuscript submitted to *Computational Linguistics*, submitted.

[6] S. Weiss and C. Kulikowski. *Computer Systems that Learn.* Morgan Kaufmann, San Mateo, CA, 1991.

[7] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.

[8] W. Daelemans, A. van den Bosch, and T. Weijters. IGTree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423, 1997.

[9] J. Bernstein and L. Nessly. Performance comparison of component algorithms for the phonemicization of orthography. In *Proceedings of 19th Annual Meeting of the Association for Computational Linguistics*, pages 19–21, Stanford, CA, 1981.

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

[11] S. Hunnicutt, H. Meng, S. Seneff, and V. Zue. 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*, pages 763–766, Berlin, Germany, 1993.

[12] E.L. Thorndike and I. Lorge. *The Teachers' Word Book of 30,000 Words.* Teachers' College, Columbia University, NY, 1944.

[13] R. I. Damper and K. Gustafson. Evaluating the pronunciation component of a text-to-speech system. In *Speech and Language Technology (SALT) Club Workshop on Evaluation in Speech and Language Technology*, pages 72–79, Sheffield, UK, 1997.

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

[15] S. M. Lucas and R. I. Damper. Syntactic neural networks for bi-directional text-phonetics translation. In G. Bailly, C. Benoît, and T. R. Sawallis, editors, *Talking Machines: Theories, Models and Applications*, pages 127–141. Elsevier (North-Holland), Amsterdam, The Netherlands, 1992.

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

[17] M. Y. Liberman and K. W. Church. Text analysis and word pronunciation in text-to-speech synthesis. In S. Furui and M. M. Sondhi, editors, *Advances in Speech Signal Processing*, pages 791–831. Marcel Dekker, New York, NY, 1991.

[18] N. McCulloch, M. Bedworth, and J. Bridle. NETspeak – a re-implementation of NETtalk. *Computer Speech and Language*, 2:289–301, 1987.

[19] R. van Bezooijen and L. C. W. Pols. Evaluating text-to-speech systems: Some methodological aspects. *Speech Communication*, 9:263–270, 1990.

[20] L. C. W. Pols. Quality assessment of text-to-speech synthesis by rule. In S. Furui and M. M. Sondhi, editors, *Advances in Speech Signal Processing*, pages 387–416. Marcel Dekker, New York, NY, 1991.

[21] J. P. H. van Santen. Perceptual experiments for diagnostic testing of text-to-speech systems. *Computer Speech and Language*, 7:49–100, 1993.

[22] C. Benoît, M. Grice, and V. Hazan. SUS test: A method for the assessment of text-to-speech synthesis using semantically unpredictable sentences. *Speech Communication*, 18:381–392, 1996.