ISCA Archive
http://www.isca-speech.org/archive

Fourth ISCA ITRW on
Speech Synthesis (SSW-4))
Perthshire, Scotland
August 29 - September 1, 2001

A Pronunciation-by-Analogy Module for the Festival Text-to-Speech Synthesiser

Robert I. Damper[†], *Craig Z. Stanbridge*[†] *and Yannick Marchand*[*]

[†]Department of Electronics and Computer Science,
University of Southampton,
Southampton SO17 1BJ, UK.

[*]Department of Psychology,
Dalhousie University,
Nova Scotia, Canada.

Abstract

Pronunciation by analogy (PbA) is a data-driven technique for the automatic phonemisation of text which is receiving renewed attention from workers in text-to-speech synthesis. It uses the dictionary which provides the primary source of pronunciations via direct look-up as a secondary source of information about the pronunciation of unknown words. In this paper, we provide theoretical and empirical motivations for the use of PbA, review approaches to automatic pronunciation generation by analogy, and report on the implementation of a PbA module for the Festival text-to-speech synthesiser. We have used a much larger dictionary (British English Example Pronunciation or BEEP, approximately 200,000 words) than hitherto. New results of 86.7% words correct are obtained for this dictionary on our best-performing PbA implementation. The Festival PbA module is still under development, however, and currently does less well.

1. Introduction

Modern text-to-speech (TTS) systems use look-up in a large dictionary as the primary strategy to determine the pronunciation of input words. However, it is not possible to list exhaustively all the words of a language, so that a secondary or ‘back-up’ strategy is required for words not in the system dictionary. Pronunciation by analogy (PbA) is a data-driven technique for the automatic phonemisation of text, first proposed over a decade ago by Dedina and Nusbaum [1, 2]. Although initially PbA attracted little attention, several groups around the world are now trying to develop the approach. There are good empirical and theoretical reasons to favour analogy over alternative approaches.

Recently, Damper et al. [3] published an empirical comparison of the performance of four representative approaches to automatic phonemisation on the same test dictionary (*The Teacher’s Word Book* of 16,280 words). As well as rule-based techniques, three data-driven methods were evaluated. These were pronunciation by analogy, the NETspeak neural network [4] and a tree-based method, IB1-IG [5, 6]. The rule sets studied were that of Elovitz et al. [7] and a set incorporated in a successful TTS product (identity kept anonymous for commercial reasons). Results showed that the data-driven techniques outperformed experts’ rules by a very significant margin: However, the data-driven methods require aligned text-phoneme datasets, and the alignment process is problematic. Nonetheless, it was abundantly clear that manually-derived rules should be abandoned in favour of data-driven approaches. Best translation results were obtained with PbA at approximately 72% words correct, compared to something like 26% words correct both for the Elovitz et al. rules and the more highly-developed rules incorporated in a commercial TTS system. Generally, in accordance with the findings of van den Bosch [6], the less compression there is of the dictionary data during learning, the better the performance. Thus, PbA is better than IB1-IG which, in turn, is better than NETspeak. Damper et al. considered a best word accuracy of 72% to indicate that automatic pronunciation of text is not yet a solved problem.

From the theoretical perspective, Pirrelli and Federici [8] write: “… the space of analogy is … eventually more accurate than the space of rules, as the former, but not the latter, is defined by the space of already attested base objects” (p. 855). There are, however, other theoretical advantages. Analogy does not use a fixed-size window on the input text nor commit to use of specific units (graphemes, phonemes, syllables, words …). Instead, input/output mappings are modelled together in variable-size chunks, so handling long-range dependencies easily and naturally. In particular, Daelemans, van den Bosch and Zavrel [9] write: “empirical results strongly suggest that keeping full memory of all training instances is at all times a good idea in language learning” (p. 38). Finally, the ‘no free lunch’ theorems of Wolpert and Macready [10, 11] tell us there is no general, ‘best’ method for machine-learning problems. They provide a basis for believing that we should prefer “appropriateness-to-task over uniformity of method” (cf. [12]) when selecting a learning technique, with all the evidence pointing to analogy as highly appropriate to the task of automatic phonemisation.

In this paper, we report on the further development of PbA for TTS applications. In particular, we have used a much larger dictionary than hitherto (British English Example Pronunciation or BEEP, approximately 200,000 words), and implemented a pronunciation module for the Festival public-domain TTS synthesiser. We commence with a brief review of previous work on PbA.

2. A Review of Pronunciation by Analogy

Several authors have described pronunciation by analogy systems which, in spite of fundamental similarities, also feature considerable variation. In this section, we outline some of these systems both to form a basis for understanding the present work and to illustrate the degree of variation between them.

2.1. Dedina and Nusbaum’s System

Figure 1 shows a block diagram of the PRONOUNCE system of Dedina and Nusbaum (D&N) [1, 2]. An input word to be pronounced is compared in turn to words in the dictionary (*Webster’s Pocket Dictionary*). Previously, the letters of each word had been automatically aligned (by a simple Lisp program) with the phonemes specifying the word’s pronunciation. Matching substrings are used to build a pronunciation lattice which is then traversed to find the ‘best’ pronunciation of the input. The ‘best’ path through the lattice is assessed according to a decision function with two heuristics (shortest path, if unique, or best scoring of the tied shortest paths). Note that there is no guarantee that there will always be at least one complete path for every input word: This is the *silence problem.*

D&N’s system was tested on just 70 monosyllabic pseudowords (as previously used in reading studies by Glushko [13, 14]). As an illustration, Figure 2 shows the pronunciation lattice for the pseudoword *shead*. Such a test is largely irrelevant to TTS applications: The test set is not representative of general English, either in the small number of words used or their length. Also, D&N’s reported results seem to be difficult (if not impossible) to replicate [15, 16, 17].

2.2. Sullivan and Damper’s System

Sullivan and Damper [18] describe a PbA system for English and German. They used a more principled alignment procedure than D&N and a different kind of pronunciation lattice in which nodes correspond to the junctures between letters, rather than to the letters themselves. (The advantage of this is that the system is never ‘silent’—unable to produce a pronunciation.) Scoring of the candidate pronunciations used something much closer to a probabilistic model than D&N’s heuristics. However, testing was again on small sets of pseudowords.

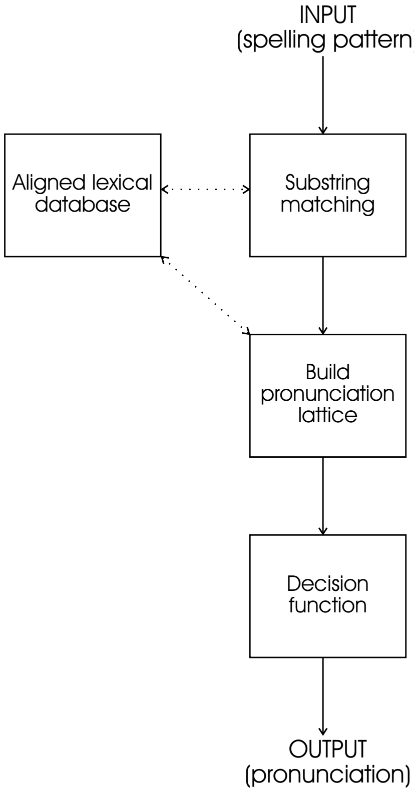

Figure 1: Block diagram of Dedina and Nusbaum’s PRONOUNCE.

2.3. Damper and Eastmond’s System

Damper and Eastmond [15, 16] reimplemented D&N’s system but improved the heuristics used to find the best path through the pronunciation lattice. Also, they conducted more realistic and relevant testing on reasonable number (some 16,000–20,000) of real words. This was done by removing each word in turn from the lexicon and deriving a pronunciation by analogy with the remaining words. (This is the n-fold cross-validation technique [19], as used by van den Bosch [6, p. 54] in the evaluation of text-to-phoneme subsystems, with $n = L$ where L is the size of the entire lexicon.) They proposed a method for silence avoidance based on Sullivan and Damper’s type of lattice, but much larger lattices resulted and, when using a large test corpus, these could not be searched in reasonable time. No analysis of errors was given. Such analysis is potentially useful in pinpointing and avoiding common sources of error.

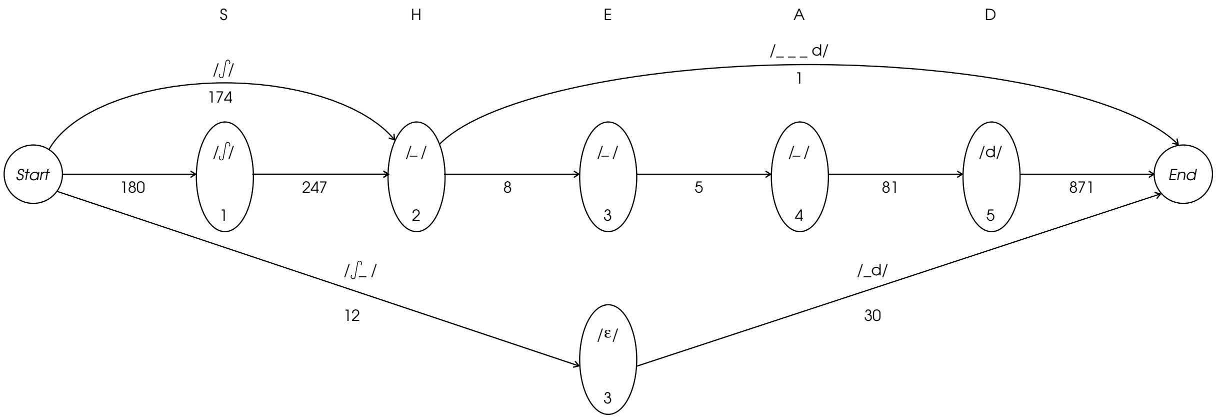

Figure 2: Pronunciation lattice for the pseudoword *shead*, simplified to show only a representative subset of nodes and arcs. The silence problem arises when there is no complete path from *Start* to *End* nodes.

2.4. Yvon's System

Recognising that paths through the D&N type of pronunciation lattice imply a (single) shared phoneme between contiguous matching substrings, Yvon [17] generalised this to produce a system (SMPA) based on "unbounded overlapping chunks". Although this was only tested on relatively small subsets (about a tenth) of the lexicon, clear performance improvements were obtained over D&N.

In a later paper, Yvon [20] develops a more formal and linguistic notion of 'analogy'. However, a system based on this notion performs less well than SMPA and is afflicted by a high silence rate.

2.5. Marchand and Damper's System

Since the principal dimension on which these different systems vary is the heuristic(s) used to score candidate pronunciations when there are tied shortest paths in the pronunciation lattice, Marchand and Damper [21] experimented with the use of multiple, simultaneous strategies, using concepts of information fusion to produce an overall decision. The five scoring strategies used to rank-order the candidates were:

1. the maximum product of the arc frequencies along the tied shortest paths;

2. the minimum standard deviation of the arc lengths along the shortest paths;

3. the maximum frequency of the same pronunciation within the shortest paths;

4. the minimum number of different phonemes between a pronunciation and the other candidates;

5. the maximum 'weak link' value, where the weak link is the minimum of the arc frequencies. (This is effectively a minimax criterion.)

A fixed number of points was then distributed among the candidates according to their position in the ranking. Individual points were then multiplied together to produce a final overall score and the best-scoring pronunciation selected as the output. Marchand and Damper showed that this multi-strategy approach gave statistically significant performance improvements over simpler versions of PbA. Further, there was a statistically significant trend for performance to improve with the number of strategies employed, and their best result was obtained using all five strategies. This best result was 65.5% words correct for the approximately 20,000 manually-aligned words of *Webster's Pocket Dictionary* (as used by Sejnowski and Rosenberg [22] to train the NETtalk neural network). This compared with a figure of 63.0% for the best-performing single scoring strategy (Strategy 3, frequency of the same pronunciation).

Marchand and Damper also used some simple heuristics for silence avoidance, with 'full' pattern matching between input letter string and dictionary entries, as opposed to D&N's 'partial' matching (see [23] for details). This is the version of PbA used in the comparative evaluation of Damper et al. [3] mentioned above, and which scored 71.8% words correct on the slightly-smaller *Teacher's Word Book*.

3. The BEEP Dictionary

One of the main aims of the present work was to use a much larger dictionary than previously. Accordingly, we have used the British English Example Pronunciation (BEEP) dictionary, available as file `beep.tar.gz` from `ftp://svr-ftp.eng.cam.ac.uk/` by following path `comp.speech/dictionaries/`.

The number of words in BEEP was initially 256,999. Of these, 70,039 words with multiple pronunciations were removed. This was done because such words are obviously problematic for PbA, but also for consistency with our previous work. This left a final total of 186,960 words with an unequivocal pronunciation. Hence, the dictionary used here is about 10 times bigger than Webster's (the 'NETtalk' corpus).

BEEP uses a set of 45 phonemes for specifying canonical pronunciations. Some of phonemes have one-character codes and some have two-character codes. For simplicity, the dictionary was preprocessed to use one-character codes uniquely.

4. Aligning the BEEP Dictionary

In common with other data-driven techniques for automatic pronunciation, PbA requires a dataset in which each letter of each word's pronunciation is aligned with a corresponding phoneme. While it is not clear that the alignment process is linguistically well-motivated, it is nonetheless unavoidable (e.g., [24]). Further, with such a large dictionary, we felt that automatic alignment was highly desirable.

The automatic technique used here is modelled on that described by Luk and Damper [25]. It is intensely ignorance-based, using an absolute minimum of prior knowledge about word spellings and their pronunciations. As such, it can be distinguished from the technique described by Black, Lenzo and Pagel [26] which requires prior specification of "allowables"—i.e., pairs of letters of phonemes which are allowed to correspond.

An association matrix $\mathbf{A}$ (initially empty) is set up with $L = 26$ rows and $P = 45$ columns, where L and P are the number of letter of phoneme classes, respectively. Processing each word of the dictionary in turn, every time a letter l and a phoneme p appear in the same word *irrespective of relative position*, the corresponding element a_{lp} of $\mathbf{A}$ is incremented. After the first pass through the dictionary, a_{lp} contains a count of the number of times letter l and phoneme p appear in the same word. An initial alignment can now be attempted.

If the numbers of letters l_w and phonemes p_w in word w differ, $n = |l_w - p_w|$ nulls are added to the shorter of the spelling or the pronunciation to make them the same length, equal to $I = \max[l_w, p_w]$. These are placed in all possible positions, producing a set of $C = \frac{I!}{(I-n)!n!}$ candidate alignments, $\mathcal{A} = \{A(c, w)\}$, $1 \le c \le C$. Each candidate alignment is scored by taking the product of the associations at each position index i:

$$S[A(c,w)] = \prod_i a_{lp}(i,c,w) \qquad \left\{ \begin{array}{l} A(c,w) \in \mathcal{A} \\ 1 \le c \le C \\ 1 \le i \le I \end{array} \right.$$

and that which maximises this score is chosen as the alignment of word w:

$$A(k,w): \quad k = \arg\max_c \left[S[A(c,w)]\right]$$

We can now perform a second pass through the dictionary to produce a new association matrix $\mathbf{A}'$ with elements a'_{lp} which count the number of times letter l and phoneme p appear *at the same (aligned) position* i. Proceeding as before, a new set of candidate alignments can be produced and scored, and a new 'best' alignment again selected. This done, further iterations can be used in an attempt to improve the alignments.

By its use of a step in which expectations of new correspondences are computed (using the current estimate of the correspondences conditioned on the dictionary data) followed by a maximisation step, this can be seen as an application of the expectation-maximisation (EM) algorithm [27, 28]. At this stage, we use just two iterations to produce $\mathbf{A}''$ from which the final alignments are obtained. The restriction to two iterations is a computational convenience; future work will use more.

5. Results

Results are given here for two different implementations of PbA using the automatically-aligned BEEP dictionary. The first is a research version of the Marchand and Damper multi-strategy analogy model written in Python [21]. This was not the version included in Festival, but is considered here because it gives a useful impression of the upper limit on pronunciation performance, not least because it has been extensively validated and debugged. The second set of results is for the simpler C++ version actually included in Festival. It implements the Damper and Eastmond analogy model with total-product scoring [16]. It has not yet been as extensively tested and developed as the Python implementation.

Pronunciation performance was assessed by n-fold cross-validation. It is worth noting that the introduction of null letters in aligning the BEEP dictionary (Section 4) actually simplifies the problem of automatic pronunciation generation slightly. Since the input to PbA contains the nulls in the appropriate places, the length of the corresponding phoneme string is also known (albeit it might also contain nulls). This is obviously important knowledge about the structure of the word's pronunciation which would not be present in a real TTS application. Ideally one should start with only the letters and guess how many nulls there should be and where they should be placed.

Using Marchand and Damper's multi-strategy model (i.e., the Python implementation not included in Festival), we obtained 85.1% words correct using all five strategies. Using Strategies 1 and 3 only, however, we obtained 86.7% words correct. It is noteworthy that better results are obtained for BEEP than for the *Teacher's Word Book* (see [21]) even though the latter is considerably smaller. This is probably because BEEP is much less diverse, containing fairly full listings of (similar) morphological derivates of stem words. Also, the present work differs from [21] in that best results are obtained using only two out of the five scoring strategies, rather than all

Rank	Letter	Errors (%)	Phoneme	Errors (%)
1	*e*	19.8	/_/	14.6
2	*a*	15.8	/ə/	12.8
3	*o*	10.7	/ɪ/	7.3
4	*r*	10.3	/z/	5.0
5	*i*	9.4	/ɛ/	4.9

Table 1: Rank ordering of letters and phonemes most often implicated in erroneous pronunciations.

of them. The reasons for this are currently unknown.

For the C++ implementation of the Damper and Eastmond model actually included in Festival, because of time constraints, we only have results for 10,431 words at this stage. This scored 57.0% words correct—much lower than the Python implementation. Some of this is undoubtedly due to the less powerful analogy model, but it is also likely that the C++ code needs further testing and improvement.

What sort of errors does PbA make? Because we only have complete results for the Python version, we restrict our attention here to this implementation. Table 1 shows in rank order the 5 letters and 5 phonemes most often implicated in erroneous pronunciations. Clearly, and not surprisingly, the vowel letters are difficult to transcribe, and this is especially true of *e*. Again not surprisingly, the null phoneme is especially problematic as is schwa, and some other vowels. (Null letters are not a great source of error as there are very few of them.) The 5 letters listed together account for 66% of all errors, while the total for the 5 phonemes is 44.6%.

6. Integrating PbA into Festival

To make the results of our work available to the wider research community, we have produced a pronunciation-by-analogy module for the Festival TTS synthesiser. Festival is a modular, extensible, multi-lingual system. Its modular design allows developers easily to integrate their own modules within the overall system. As it provides a complete, existing synthesis environment, allowing us to concentrate solely on the pronunciation module, it is ideal for our requirements. Festival is public-domain for research purposes, and is available for download from `http://www.cstr.ed.ac.uk/`.

The default way of producing a pronunciation for an unknown word in Festival uses the (learned) letter-to-sound rules as described by [26]. Festival provides a function for specifying what module is to be used for this process, allowing for custom modules to used instead. The only requirements of the new module are for it to take the unknown word and any features (i.e., items such as part of speech and stress patterns—we do not use features at this stage). Integrating the source code of the PbA module was simply a case of placing the C++ source code into the appropriate module directory, in this case `/modules/Lexicon/`, declaring the function prototypes in file `lexicon.cc` in this directory, and making the appropriate changes to the `makefile`. A call to the function `(lex.set.lts.method ’pba.lookup)` then set Festival to use our PbA module. From this point on, all unknown words—absent from the BEEP dictionary—are passed to the PbA module.

7. Conclusions

Pronunciation by analogy (PbA) is a powerful technique for automatic phonemisation in text-to-speech (TTS) synthesis. We have described recent developments in PbA and the implementation of a pronunciation module for the Festival public-domain TTS synthesiser using the BEEP dictionary as a database. This module is currently undergoing development and improvement to bring its performance closer to that of our reference, research implementation. Future work is planned to use more iterations (than the present two) of the EM algorithm in automatic alignment of the dictionary, to investigate the convergence of the alignment algorithm, and to perform both objective and subjective comparison of PbA with Festival's default pronunciation module (the learned letter-to-sound rules of Black, Lenzo and Pagel.)

8. References

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

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

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

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

[5] W. Daelemans, A. van den Bosch, and T. Weijters. IGTree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11(1–5):407–423, 1997.

[6] A. van den Bosch. *Learning to Pronounce Written Words: A Study in Inductive Language Learning*.

PhD thesis, University of Maastricht, The Netherlands, 1997.

[7] 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 Speech and Audio Processing*, ASSP-24(6):446–459, 1976.

[8] V. Pirrelli and S. Federici. You'd better say nothing than something wrong: Analogy, accuracy and text-to-speech applications. In *Proceedings of 4th European Conference on Speech Communication and Technology, Eurospeech'95*, volume 1, pages 855–858, Madrid, Spain, 1995.

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

[10] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM, 1995.

[11] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[12] J. van Santen and R. Sproat. Introduction. In R. Sproat, editor, *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*, pages 1–6. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[13] R. J. Glushko. The organization and activation of orthographic knowledge in reading aloud. *Journal of Experimental Psychology: Human Perception and Performance*, 5(4):674–691, 1979.

[14] R. J. Glushko. Principles for pronouncing print: The psychology of phonography. In A. M. Lesgold and C. A. Perfetti, editors, *Interactive Processes in Reading*, pages 61–84. Lawrence Erlbaum Associates, Hillsdale, NJ, 1981.

[15] R. I. Damper and J. F. G. Eastmond. Pronouncing text by analogy. In *Proceedings of 16th International Conference on Computational Linguistics*, volume 2, pages 268–273, Copenhagen, Denmark, 1996.

[16] 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.

[17] F. Yvon. Grapheme-to-phoneme conversion using multiple unbounded overlapping chunks. In *Proceedings of Conference on New Methods in Natural Language Processing (NeMLaP-2'96)*, pages 218–228, Ankara, Turkey, 1996b.

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

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

[20] F. Yvon. Paradigmatic cascades: A linguistically sound model of pronunciation by analogy. In *Proceedings of 35th Annual Meeting of the Association for Computational Linguistics*, pages 429–435, Madrid, Spain, 1997.

[21] Y. Marchand and R. I. Damper. A multi-strategy approach to improving pronunciation by analogy. *Computational Linguistics*, 26(2):195–219, 2000.

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

[23] R. I. Damper and Y. Marchand. Improving pronunciation by analogy for text-to-speech applications. In *Proceedings of 3rd European Speech Communication Association (ESCA)/COCOSDA International Workshop on Speech Synthesis*, pages 65–70, Jenolan Caves, Australia, 1998.

[24] R. I. Damper. Learning about speech from data: Beyond NETtalk. In R. I. Damper, editor, *Data-Driven Methods in Speech Synthesis*. Kluwer Academic Publishers, Dordrecht, The Netherlands, forthcoming.

[25] R. W. P. Luk and R. I. Damper. A novel approach to inferring letter-phoneme correspondences. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'91*, volume 2, pages 741–744, Toronto, Canada, 1991.

[26] A. W. Black, K. Lenzo, and V. Pagel. Issues in building general letter-to-sound rules. In *Proceedings of 3rd European Speech Communication Association (ESCA)/COCOSDA International Workshop on Speech Synthesis*, pages 77–80, Jenolan Caves, Australia, 1998.

[27] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistics Society, Series B*, 39(1):1–38, 1977.

[28] T. K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, November 1996.