Chapter for inclusion in *Connectionist Models of Memory and Language*, edited by J. Levy, D. Bairaktaris, J. Bullinaria and P. Cairns.

Self-Learning and Connectionist Approaches to Text-Phoneme Conversion

R.I. Damper,
Image, Speech and Intelligent Systems (ISIS) Group,
Department of Electronics and Computer Science,
University of Southampton,
Southampton SO17 1BJ,
UK.

1 Introduction

The automatic derivation of the pronunciation of an English word from its spelling is a difficult problem of some practical significance. Most often, pronunciation will be specified as an idealised phonemic 'baseform' (which may or may not include stress markers) so that we refer here to this process as *text-phoneme conversion*.

Interest in the conversion problem comes from two rather different points of view. First, interactive computer systems featuring speech output generally have a requirement for textual input. It makes sense to translate this input to some intermediate representation (e.g. phonemic) closer to the actual sounds to be synthesised. Second, the conversion process is a key component of many models of human language processing – particularly reading aloud. The earliest attempts at automated conversion (Ainsworth, 1973) were made by speech scientists concerned with the technology of synthesis. These avoided dictionary matching (because of limitations of computer memory) and were based on a set of letter-to-phoneme rules, manually written by expert phoneticians to capture the regularities which clearly exist in spelling-sound correspondence. This formalism remains popular in both fields of endeavour. For instance, present-day commercial text-to-speech (TTS) systems achieve acceptable performance by combining dictionary matching with the use of rules for the translation of words not present in the dictionary. Apart from this commonality in the use of a rule-based formalism, and given that essentially the same process is under study, it is perhaps surprising that the two fields of speech synthesis and psychological modelling have not informed one another more than they have so far done.

In both areas, however, there is currently much interest in inductive learning of the regularities of text-phoneme correspondence. If the automatic, conversion process is also *self-learning*, then this should reduce the manual effort necessary to build a TTS system. Further, since the human ability to read is acquired rather than innate, a self-learning converter is better able to serve as a model of language processing.

Recently, connectionism has emerged as an influential paradigm in psychology, as well as figuring prominently in speech technology, largely stemming from the discovery of powerful self-learning algorithms such as error back-propagation (Rumelhart *et al*, 1986). From the psychological perspective, the parallel distributed nature of a connectionist model lends it a good deal more plausibility than a set of rules since, unlike the latter, we can readily see how the former might be implemented in 'brainware'. Quite apart from any concern with TTS systems or the modelling of human language processing, text-phoneme conversion has proved a popular application for workers interested in neural computation for its own sake. Because the mapping between text and phonemes is complex, the problem provides a good test for the power of neural solutions, and has become something of a standard in this respect.

The literature on connectionist and self-learning approaches to text-phoneme conversion is considerable but widely dispersed. In particular, as stated above, work in the area of speech synthesis has not always been informed by relevant work in psychological modelling, and *vice versa*. In this chapter, we present a critical, unifying review of this literature which we believe to be the first such to appear. As well as collecting the various pieces of work together so as to render them more accessible, we also categorise approaches and highlight relations between them. This is done to form a framework for the following two chapters, as well as in introduction to them. Not all the approaches dealt with are obviously connectionist in inspiration; they are included for completeness and because, in some cases, a rather obvious connectionist implementation is possible – if not necessarily parsimonious or efficient.

The chapter is structured as follows. Section 2 outlines the traditional, rule-based approach to text-phoneme conversion. Subsequently, a variety of automatic discovery techniques is treated in section 3, namely generate-and-test rule induction, rule induction by clustering, decision-tree induction, Markov modelling, back-propagation networks in general, synthesis-by-analogy (including memory-based reasoning) and syntactic neural networks. Where the techniques are not neural in inspiration or in origin (e.g. decision-tree induction and Markov modelling), prospects for neural implementation are considered. Finally, section 4 summarises.

2 Principles of Rule-Based Translation

The synthesis of unrestricted-vocabulary speech from orthographic text is, in the words of Klatt (1987, p. 781), "a new technology with a rapidly changing set of capabilities and potential applications." Present-day TTS systems typically use a large dictionary of pronunciations in conjunction with a set of letter-to-sound rules to produce a phonemic transcription of the text (Allen, 1976; Allen *et al*, 1987). The rules are invoked to transcribe words for which no dictionary match is found.

For English at least, the text-phoneme conversion process is far from trivial, reflecting the many complex historical influences on the spelling system (Venezky, 1965; Scragg, 1975). Indeed, Abercrombie (1981, p. 209) describes English orthography as "... one of the least successful applications of the Roman alphabet." Some of its well known vagaries are that letter combinations (*ch, gh, ll, ea*) frequently act as a unit (a 'grapheme') signaling a single phoneme, a single letter occasionally corresponds to more than one phoneme (as in (*six*, /sɪks/)), pronunciation can depend upon word class (e.g. *convict*, *subject*) and there can be non-contiguous "markings" as with the final, mute *e* of (*make*, /meɪk/) (Wijk, 1966; Venezky, 1970).

Ever since the pioneering work of Ainsworth (1973), text-phoneme conversion for speech synthesis has (in the absence of a dictionary-derived pronunciation) traditionally used a context-dependent rewrite rule formalism of the sort favoured in generative phonology (e.g. Chomsky and Halle, 1968, p. 14). Such rules can also be straightforwardly cast in the *IF... THEN* form commonly employed in expert systems technology. The notion underlying the conversion process is that it is possible to arrive at a translation for any "letter unit" (grapheme) – and thereby for a whole word – provided enough contextual information is available. Rules are of the form:

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

which states that letter substring B with the left context A and right context C rewrites to phoneme substring D. Influential rule sets in this tradition are those of Elovitz *et al* (1976) and Hunnicutt (1976).

Because of the complexities of English spelling-to-sound correspondence, 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 sub-lists, grouped by (initial) letter and with each sub-list ordered by specificity. Typically, the most specific rule is at the top and most general at the bottom. In the Elovitz *et al* rules, 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 sub-list 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 sub-list is a context-independent *default* for the target letter, which is fired in the case that no other, more specific rule applies. Transcription is rather more complex with the Hunnicutt rules which use three passes; also, processing can be in either direction. First, affixes are stripped, then consonants are converted, and finally vowels and affixes are transcribed. This procedure allows converted (phonemic) strings which are highly dependable (i.e. for the consonants) to be used as context in the more complex rules for vowel and affix transcription. Again, rules are ordered to facilitate conflict resolution; however, the determination of an appropriate ordering is often problematic (Carlson *et al*, 1990, p. 275). After conversion of letters to phonemes, lexical stress can be added using a similar context-dependent formalism; see Church (1985) for a review.

In spite of obvious commonality between the computational process of TTS conversion and the psychological process of reading aloud, there has been very little interaction between techniques for the former and models of the latter. As we have just seen in the case of speech synthesis, the standard approach utilises a pronouncing dictionary for known words and a set of general-purpose, context-dependent translation (CDT) rules which is invoked if the input word is not in the system's dictionary. In the case of reading aloud, the standard model also involves two routes to pronunciation. So-called *dual-route theory* (e.g. Forster and Chambers, 1973; Coltheart, 1978) posits a lexical route for the pronunciation of known words and a parallel, simultaneously-activated route utilising abstract grapheme-phoneme conversion (GPC) rules for the pronunciation of unknown, or 'novel', words (Figure 1). Unlike the sequential application of dictionary matching followed by contingent rule-based translation in a TTS system, the two routes are usually conceived as operating essentially in parallel in the psychological model. Arguments for dual-route theory are based on the ability to pronounce pseudowords, latency difference effects between regular and exception words, and apparent double dissociation between the

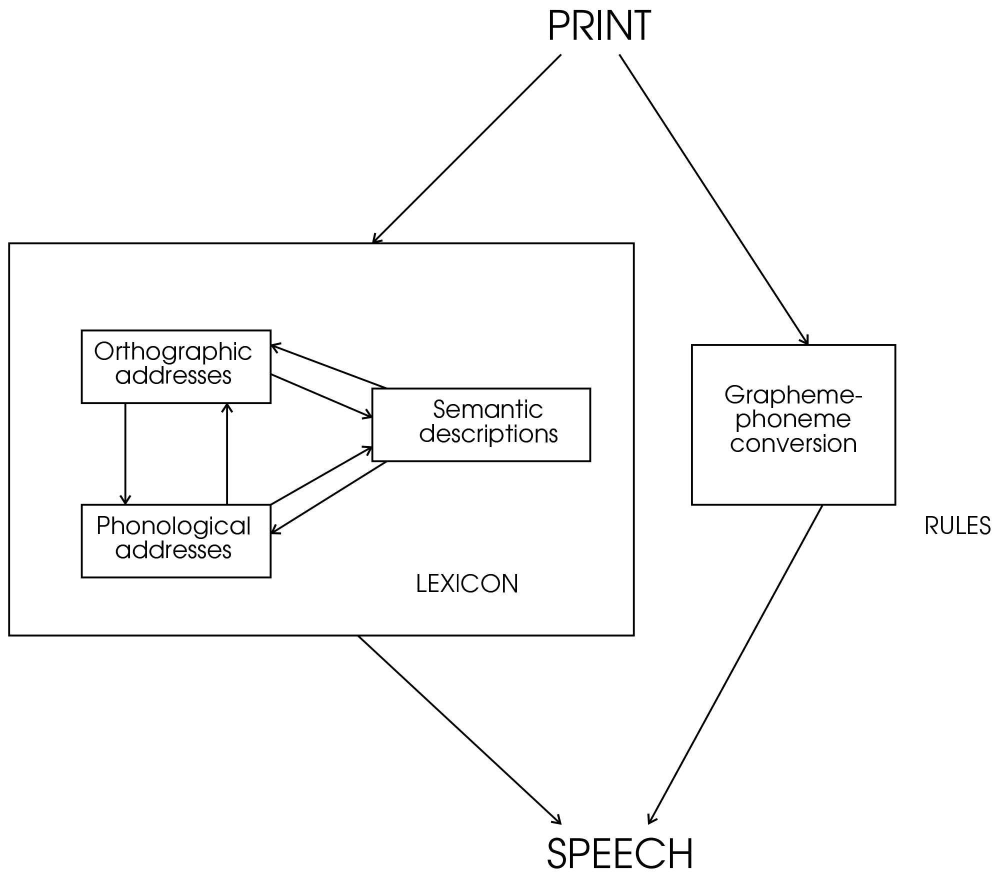

Figure 1: Schematic of the dual-route model for the production of pronunciation from print. *After Humphreys and Evett (1985).*

two routes in dyslexia (see Humphreys and Evett, 1985, for an extensive review and discussion).

It follows from earlier discussion of the complexities of letter-sound correspondence in English that the task of writing an adequate set of CDT rules is labour-intensive and requires detailed, expert knowledge of the particular language. Once the rule set reaches a certain size, it becomes difficult to modify in the light of errors, because of the potential for interaction between rules. Problems such as this have led to the production of rule compilers (Carlson and Granström, 1975; Hertz *et al*, 1985; Van Leeuwen, 1989) which aim to give automatic assistance to rule developers for speech synthesis systems. Clearly, the necessity for such devices reduces the psychological plausibility of the rule-based approach greatly: it is difficult to imagine humans employing the mechanisms embodied in rule compilers for language acquisition. Recently, however, interesting advances have been made in applying self-learning – or 'automatic discovery' – techniques to the problem of mapping text to phonemes. Not only does this offer the promise that much of the labour in traditional rule generation might be avoided, but the techniques are much more reasonable from a psychological point of view as they are well suited to implementation in 'brainware'.

In this approach, transcribed texts or entries in a machine-readable pronouncing dictionary are treated as sources of training data from which the system learns generalisations useful for the transcription of seen and unseen words alike.

3 Review of Self-Learning Systems

In this section, we review previous attempts to generate automatically systems capable of performing text-phoneme transcription.

3.1 Generate-and-test rule induction

Early work applying self-learning techniques to text-phoneme conversion (Oakey and Cawthorne, 1981; Klatt and Shipman, 1982) retained the CDT rule formalism and aimed to infer a rule set by a generate-and-test process. These attempts were not inspired by connectionist ideas: they are included here for completeness.

Oakey and Cawthorne started from a context-independent base set (the default rules) and worked through the dictionary, generating pronunciations for each word and comparing this with the known, correct version. Any difference between generated and correct pronunciation was then used to create a new, special-purpose rule to cater for the mispronunciation. This creation process requires an *alignment* of text and phonemes, i.e. we need to know which letter(s) to associate with which phoneme. Oakey and Cawthorne's technique for this appears rather *ad hoc*, being based on a "look-ahead" heuristic. Subsequently, any such special-purpose rules which were sufficiently similar were generalised by combination to produce somewhat more general rules, which were themselves candidates for combination. Like the alignment technique, the combination methodology looks fraught with problems. Typical questions which arise during generate-and-test rule inferencing include: Are the problematic rules too general or too specific? Should the left context, the right context or both be adjusted? Should the target substring be increased or reduced in length?

Klatt and Shipman (1982), for instance, attempted to avoid such problems by selecting "only the most popular of each set of conflicting rules"; these were then formed into a decision tree to facilitate rapid translation. Although they do not give performance figures or make any subsequent report on their work, Klatt later states (1987) that an error rate of 7% was obtained. The rules were inferred from 10,000 words obtained by randomly dividing a 20,000-word pronouncing dictionary in two, and the system was tested on the unseen half of the dictionary. Such a high level of performance is remarkable in view of the gross way of treating conflicting rules.

Oakey and Cawthorne tested their set of rules on the training data after deleting all "basic" rules, i.e. the created rules which (by their very nature) were entirely specialised to the mispronunciations. Of course, if this had not been done, performance on the training data would have been 100%. In this way, the generalisation power of the technique was under test (although not exactly on 'unseen' words). On the Ladybird Key Word texts (477 words) for beginning readers, scores of 84% phonemes correct and 59% words correct were obtained. Corresponding figures for the Elovitz *et al* rules, manually anglicised, were 91% and 84% respectively. Clearly, this is not a very demanding data set. On the 1015 words in their pronouncing dictionary with initial letter *a*, however, the relevant figures were 74% phonemes correct and 21% words correct. This was actually superior to the respective figures of 64% and 16% obtained with the modified Elovitz *et al* rules.

Van Coile (1990) describes a generate-and-test rule induction process which appears to be a significant improvement over that of Oakey and Cawthorne. Rules are generated for one letter at a time. Starting with aligned data (see section 3.4 below), the training set is searched for occurrences of that letter, and each occurrence furnishes a pronunciation example. The left and right contexts are subject to an upper limit of 3 letters each. The phoneme that most frequently occurs in the examples is taken as the right-hand side of the (context-independent) default rule. Initially, pronunciation examples handled correctly by the default are marked *realised* and the remainder are marked *not yet realised*. The creation of new rules then proceeds iteratively. All possible different letter contexts are generated for the *not yet realised* examples. Next, one phoneme is associated with each generated context and the rule (i.e. context plus phoneme) that most improves the performance of the current rule set is added to the current set. The examples are now scanned again and either marked *realised* or *not yet realised*. (Van Coile notes that it is possible for *realised* examples to be re-marked *not yet realised*.)

The induction process is kept tractable by considering contexts of one fixed length, C, at a time in conjunction with two other parameters, M and T. M is the maximum context length that occurs in the previously-determined rules and T is a performance-increase threshold which must be exceeded for a newly-generated rule to be added to the current set. After learning on a random selection of 7000 of the 10,000 most common Dutch words, which generated an average of 14 rules per letter, the method was evaluated on the remaining 3000 words. Performance "at the grapheme level was better than 96%" which, ignoring stress, corresponds to roughly 82% correct at word level. Van Coile concedes that this is inferior to the performance obtained from traditionally-developed rule sets (but see discussion above concerning the difficulty of assessing rules alone) yet believes the inductive technique is "very useful to obtain a good initial rule set for further manual development".

A question worth posing is: why does Van Coile's generate-and-test approach to rule induction produce so much better results than Oakey and Cawthorne's initial attempt? It seems likely that the key factors are the use of a principled alignment technique based on the Viterbi algorithm (see 3.4 below), together with the strategy of processing one letter at a time across the whole training set, rather than processing one word at a time. It is difficult to say in quantitative terms how much easier the transcription task is for Dutch than for English.

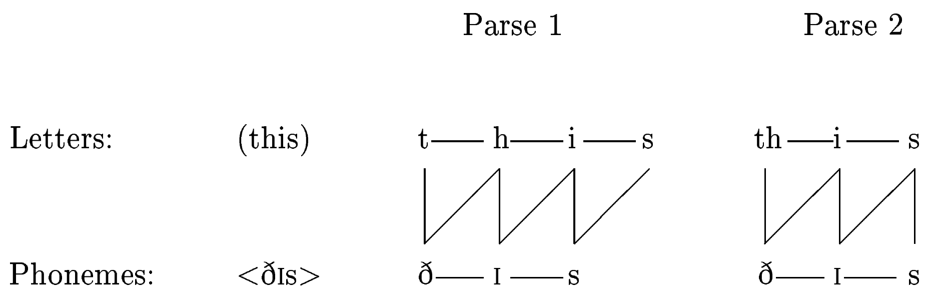

Figure 2: Contiguous symbol clusters in the letter (L) and phoneme (P) domains. Contiguities, within domains and between them, are shown with connecting lines. After (hypothetical) Parse 1, the letter cluster *th* is recognised as a common substring, allowing the LP cluster (*th*)<ð> to be identified at Parse 2. *After Wolff (1984).*

3.2 Rule induction by clustering

Wolff (1984) describes a method of inducing CDT rules based entirely on clustering. While he calls his formalism "context-free" (p. 12) the end result is a *concatenation* of context-independent rules, which amount to context-dependent rules.

The method first searches the dictionary (in multiple passes) looking for common clusters of contiguous symbols "using frequency and size of cluster as a guide to which amongst the manifold alternatives are best". Clustering occurs not only within the separate domains of letters (denoted L) and of phonemes (denoted P), but across domains also. This demands an extension of the concept of *contiguity* to the cross-domain (LP) situation. Wolff's solution is to parse the orthographic and phonemic representations "in step", and to consider as contiguous those elements (i.e. single letters, single phonemes or clusters) which are identified in sequence.

Figure 2 illustrates this for the example word (*this*, /ðɪs/), where all contiguities are shown as connecting lines. Initially (Parse 1) then, letter t is considered contiguous not only with letter h in the L domain but with phoneme /ð/ in the P domain also. Imagine that after the first pass through the dictionary, the L cluster *th* is identified. Then, at Parse 2, *th* is now considered to be contiguous both with i and /ð/. Similarly, both letter i and phoneme /ɪ/ are considered to 'follow' /ð/. In this way, Wolff aims to build up LP clusters such as (*th*)<ð>, which function as context-independent rules or, in Wolff's terminology, "basic" rules. Note how the clustering technique involves an implicit alignment of text and phonemes, rather than requiring an explicit, prior alignment.

Clearly, the set of "basic rules" (being context-independent) will in general be ambiguous. For instance, both (*th*)<ð> and (*th*)<θ> are likely to exist and there is no way of telling (without further analysis of the training data) which should be used in translating a given occurrence of *th*. Wolff uses a method of disambiguation which is a variant of the clustering process. Having first identified ambiguous rules, contiguous pairs of such rules are clustered to produce new (context-dependent) "complex" rules. This should continue iteratively until no ambiguity remains in the rule set.

In the implementation reported, only one "rule" (cluster) is formed for each pass through the dictionary – clearly a very inefficient scheme. Unfortunately too, due to a software bug, Wolff was unable to run his program to completion. Nonetheless, interesting results were obtained. With a dictionary of 800 (most-frequent) words, the first *LP* cluster formed was (*t*)<t>. Other early *LP* clusters to emerge were (*er*)<ə> and (*ng*)<ŋ>. Since a complete set of "basic" rules could not be obtained automatically, the disambiguation process was tested on 252 hand-compiled "basic" rules. "Complex" rules were automatically formed up to an (arbitrary) limit of 200 in number, and their inclusion shown to reduce the size of the ambiguity set. Given the promise of this approach, it is a pity that a complete working system was never achieved.

Although Wolff's work was not obviously inspired by connectionism, clustering is a fundamental property of many unsupervised neural schemes so that a connectionist implementation should not be difficult.

3.3 Decision-tree induction

Segre *et al* (1983) describe a system using "two transcription rule sets which are machine generated over a set of sample transcriptions". The first operates on words and produces stress assignments; the second subsequently maps a letter "cluster" (with known stress value) onto its corresponding phoneme string. The approach of assigning lexical stress first is an interesting departure from the usual procedure. These authors base the rule-inference process on machine-learning techniques like that of Quinlan (1979; 1990), which generate decision trees based on sets of examples and their classifications, in terms of an adequate set of *features*. Although they give a step-by-step description of the inference algorithm as applied to the generation of the stress rules, the learning of phonemic transcription rules is effectively ignored (beyond saying it is "similar"). In particular, the way the letter clusters are formed is not stated. Disappointingly, no performance data are given.

Lucassen and Mercer (1984), however, describe what was arguably the first self-learning technique to be theoretically well-founded and to achieve a (quantified) measure of success. Accordingly, it has materially influenced later approaches and so we describe it in some detail here.

Lucassen and Mercer view the transcription process as a transformation from a word's spelling, s, to its phonemic baseform, β, via a noisy channel. The term "phonemic baseform" describes the (idealised) word pronunciation as might be found in a pronouncing dictionary. It is used by Lucassen and Mercer in distinction to "pronunciation" which, in their terminology, is the phoneme or phonemes corresponding to a single letter. The current (target) letter has a context of letters to left and right and of phonemes to the left; all these symbols together constitute the *channel context*.

The *features* of the model determine a partition either of the set of letters, L, or of the set of phonemes, P. The pronunciation π of the current letter is thus a string which is an element of the power-set P^*. The authors point out the similarity of this formalism to a set of context-dependent rules (except that probabilities are involved here).

Lucassen and Mercer aim then to construct a vector, $\mathbf{h}$, of binary features for channel contexts consisting of the current letter (L_0), the 4 letters to the left and the 4 letters to the right of L_0 ($L_{-4}, \ldots, L_4$), and the 3 phonemes to the left of L_0 ($P_{-3}, \ldots, P_{-1}$). Note the requirement for text and phonemes to be aligned since the 3 phonemes to the left of L_0 must be known. They assume that the best binary feature for one of these symbols is that which maximises the mutual information between features and pronunciations, and present a (sub-optimal) algorithm to construct this. Given the best binary feature, another (presumed close to the next-best) can be computed and included with the original to produce a two-element feature vector which effects a 4-way partition of the relevant set. This feature inclusion proceeds until a complete partition is achieved. Altogether, for their lexical data, a 6-element vector is required to give a complete partition of L, and an 8-element vector for P. The concatenation of the feature vectors for $L_{-4}, \ldots, L_4$, $P_{-3}, \ldots, P_{-1}$ corresponds to the 78-element feature vector $\mathbf{h}$ for the entire channel context. At this stage, the automatically-selected features are used to build a decision tree which determines the order in which the bits of $\mathbf{h}$ should be examined, and how many bits should be examined, before selecting a pronunciation. To each leaf of the resulting decision tree is attached a probability distribution over the pronunciations, evaluated as the weighted sum of *a posteriori* maximum likelihood estimates at each node on the path from root to leaf.

After training on their lexicon, the system was tested on 194 words chosen at random from IBM's office-correspondence database having a vocabulary of 5000 words. Of 1396 phonemes, 1308 (or 94%) were correctly transcribed. Interestingly, 47 of the 88 errors (53%) were reported to be errors of vowel stress placement. The authors do not state how many of the test words, if any, were absent from the training data. It is unfortunate that testing was on such a small data set.

The decision-tree induction approach is not connectionist in origin or inspiration. However, many workers have considered neural implementations of decision trees, or the combination of the two approaches (e.g. Stromberg *et al*, 1991; Rahim, 1994).

3.4 Markov model techniques

Like many subsequent approaches, the Lucassen and Mercer methodology is dependent on having available aligned (s, β) word pairs, in order that the i-index of P_i can be related to L_0. Alignment was achieved by recognising that that their translation formalism was a kind of hidden Markov model (HMM), so that standard algorithms could be exploited. In fact, they used for alignment a much a simplified model of the spelling-to-baseform channel in which the channel 'context' consisted solely of the current letter. The parameters of this model were optimised using the forward-backward algorithm (Baum, 1972) and a dynamic-programming technique applied to find the most probable alignment of letters with phonemes.

Thus, Lucassen and Mercer faced a 'bootstrapping' problem of alignment: to align text and phonemes effectively, as a first step in inferring correspondences between the two domains, one needs a set of correspondences (explicit or implicit) to start with. Indeed, the requirement for an explicit correspondence set is central to later alignment techniques, such as that of Lawrence and Kaye (1986) which has become a standard. In principle, for a technique to be truly self-learning, any necessity for a *prior* alignment phase (i.e. using explicit, pre-compiled correspondences) should be avoided – a point which is taken up in the following chapter by Bullinaria.

The use by Lucassen and Mercer of the hidden Markov model techniques which were then starting to dominate speech recognition has inspired a small number of more recent attempts to employ an HMM formalism for text-phoneme conversion.

Van Coile (1990), whose rule-induction work is described above, has used a hidden Markov phoneme model in conjunction with the Viterbi algorithm (Viterbi, 1967; Forney, 1973) to align the orthographic and phonemic representations of words prior to a rule-induction phase. When trained on the 10,000 most-frequent Dutch words and using 1/72 (see below) as the initial output probability for all phonemic symbols, 98.5% correct alignment of words results (relative to a manual alignment), corresponding to better than 99% correct phoneme alignment.

Van Coile's program supports 72 different output (phoneme) symbols. The use of an equiprobable initial output distribution, while having the strong virtue of minimising the assumptions made, might be expected to lead to problems with training. By contrast, initial transition probabilities appear to have been decided on the basis of intuition. They are detailed in the paper. The need to decide on some specific values for the initial probabilities is, of course, just another instance of the 'bootstrapping' problem. This fact becomes plain when Van Coile also reports alignment results using output probabilities based on "some very simple observations about the correspondence between orthographic and phonetic representations in Dutch." In general, these output probabilities gave superior results to the equiprobable values for smaller training sets, but the advantage disappeared as the size of the training set increased.

Rather than using HMM techniques for prior alignment only, Parfitt and Sharman (1991) extend the formalism to give a complete, bi-directional (text-phoneme and phoneme-text) model. Thus, in the case of predicting pronunciation from spelling, the orthography is seen as the observed sequence of output symbols emitted by the model as it makes transitions between its hidden, phonemic states. In its present form, each state corresponds to a single phoneme; thus, the Markov property dictates a simple bigram model. Parfitt and Sharman point out, however, that the formalism is trivially expandable to higher-order n-grams. The problem then is to find the (phonemic) state sequence which accounts for the observed (orthographic) output with greatest probability; this problem is solved using the Viterbi algorithm. Once obtained, initial estimates of the model's parameters (phoneme-transition probabilities and output probabilities) can be subsequently re-estimated using the forward-backward algorithm. While the initial transition probabilities were simply estimated by frequency counts using a (frequency-weighted) pronouncing dictionary, it is not so simple to estimate the initial output probabilities. Two possibilities are to assume equiprobability (as did Van Coile) or to make frequency counts of word pairs after orthographic-phonemic alignment using an algorithm like that of Lawrence and Kaye (1986). Parfitt and Sharman, however, adopt initially an intermediate position of assuming that each phoneme in a word's "phonetic" form has an equally-likely chance of generating any of the letters in the orthographic form, although subsequently they do employ a dynamic-programming alignment algorithm as well.

The model was trained on a 50,000-word spoken English corpus, which had been previously transcribed into both orthographic and phonemic form. Parfitt and Sharman do not specify the size of their training and test sets; the values which follow come from a personal communication. A 41,169-word section was selected for training, and a disjoint 1290-word section extracted for testing. There were 8149 distinct words in the training text, and 626 in the test text with an unknown number of these not present in the training data. Transcription accuracy is approximately 53% phonemes correct for initial estimates based on unaligned word pairs and with frequency counts normalised for *a priori* occurrence of phonemes. After 4 iterations of Baum-Welch re-estimation, this improves to approximately 70% phonemes correct. Using the dynamic-programming alignment improves the initial (normalised) estimates to 85% phonemes-correct.

While forward-backward re-estimation improved performance in the case of the poorer initial estimates of the model's parameters, for the better initial estimates based on alignment, performance actually deteriorates (to around 76% after 4 iterations).

Luk and Damper (1991; 1994) describe an approach to text-phoneme translation that they call *stochastic transduction*. The approach is based on formal language theory and, in particular, the use of a stochastic 'transduction' grammar to model the translation process. In this work, the terminal symbols of the transduction grammar are text-phoneme correspondences and the translation of a word is modelled as sentential derivation producing a string of such correspondences. It is self-learning in that the terminals (correspondences) are inferred from training data, as are the probabilities of the rewrite rules of the stochastic grammar. Because the work in its current form embodies a regular grammar and the Markov assumption for the rule probabilities, it is essentially a form of Markov (but not hidden) modelling. However, these restrictions are not necessary so that the stochastic transduction formalism is in principle considerably more general than HMM approaches.

Luk and Damper (1994) have reported 100% alignment performance on 18,767 training words and 1667 unseen, test words from the Oxford Advanced Learners' Dictionary (Oxford University Press, 1989). The percentage of words correctly translated is 72% for the training set and also 72% for the test set, for a version of the model in which correspondences were inferred on the presumption that they should end just after a vowel or after at most 2 consonants in the phoneme domain.

Again, the hidden Markov model formalism developed in a way that was entirely divorced from notions of connectionism. Recently, however, a number of authors has considered relations between the two (Bridle, 1990; Nádas, 1994). HMMs are generative models based on statistical distributions, and can be trained by statistical estimation procedures. On the other hand, neural networks are not usually generative nor trained by statistical methods but by gradient descent, error minimisation. These differences, however, are not irreconcilable. By considering neural models which can be trained on a maximum likelihood basis, connectionist implementations of HMMs can be realised.

3.5 Back-propagation networks

The best-known example of a self-learning text-to-phoneme system is probably NETtalk, a back-propagation network described by Sejnowski and Rosenberg (1987). This is described in some detail by Bullinaria in his following chapter, and so is only briefly dealt with here. NETtalk was primarily an attempt to model aspects of human learning. One of the most attractive characteristics of connectionist models is surely that their distributed, parallel nature lends them a degree of psychological plausibility. In what follows, we consider other work which has used multi-layer, feedforward nets trained by (supervised) error back-propagation for text-phoneme conversion.

McCulloch *et al* (1987) describe NETspeak (Figure 3), intended principally as a re-implementation of NETtalk. However, this work 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 which was stepped across the input one character at a time. As does Klatt (1987), McCulloch *et al* cite Lucassen and Mercer (1984) as having demonstrated that a 7-character window contains sufficient context to produce a reasonable transcription while avoiding the high computational expense of larger windows. However, Lucassen and Mercer actually used a *nine*-letter window. Further, the view of Church (1985, p. 246) is relevant: "... stress dependencies cannot be determined locally. It is impossible to determine the stress of a word by looking through a five or six character window."

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." Each input character was represented in NETtalk by a sparse (1-out-of-n) code of 29 bits

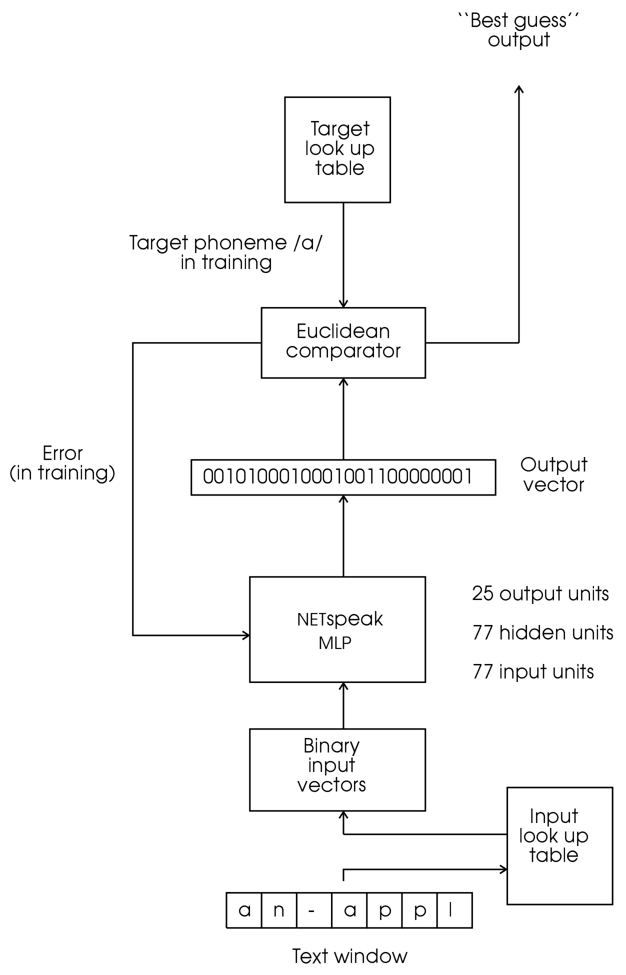

Figure 3: The architecture of NETspeak. *Redrawn from McCulloch et al (1987).*

– one bit for each of the 26 letters and 3 additional bits for punctuation marks. Thus, the number of input units, i, was $7 \times 29 = 203$. By contrast, NETspeak used a more compact (2-out-of-n) 11-bit coding, giving $i = 77$. The first 5 bits indicated which of 5 rough, "phonological sets" the letter belonged to and the remaining 6 bits identified the particular character. In place of NETtalk's 21 "articulatory features" to represent the (single) phoneme output (plus 5 stress and syllable boundary units), NETspeak used $o = 25$ output features. Denoting the number of hidden units as h, NETtalk used $h = 120$ for most experiments, whereas NETspeak used 77 hidden units. The relative numbers of neurons are thus $(203+120+26) = 349$ for NETtalk versus $(77+77+25) = 179$ for NETspeak. Since, for a fully-connected MLP with one hidden layer, the total number of interconnections is $(i \times h) + (h \times o)$, NETtalk had 27,480 adjustable weights (ignoring bias inputs to the neurons which are generally treated as adjustable weights also) compared to NETspeak's 7854.

NETspeak was trained on 15,080 of the 16,280 (previously-aligned) words in a pronouncing dictionary. Given differences of detail between the networks (e.g. in training material, in learning parameters and in evaluating the "best guess" output), a direct comparison with NETtalk is difficult. However, NETspeak achieved 87.9% "best guess" performance on these 15,080 words after some 3 passes through its training set. Further, performance on new (unseen) words was only slightly worse at 86% indicating a high degree of generalisation of the regularities of text-phoneme correspondence. The authors do not explicitly state the size of this test set. Presumably, it consisted of the $(16,280 - 15,080 = 1200)$ held-out words.

The net was subsequently trained on a frequency-weighted version of the dictionary. The authors expected that the network would have more difficulty learning this training set, because of the higher proportion of commoner (and presumably, therefore, irregularly-pronounced) words. This turned out not to be the case although performance on common words was reported to improve while that for "very regular words" deteriorated.

Further, McCulloch *et al* trained two separate networks on common and on uncommon words. In the former case, the training set was 13,021 non-unique words representing in appropriate (frequency-weighted) proportions a randomly-chosen 958 of the 1058 most common words. The test set consisted of 911 words representing the remaining 100 words in appropriate proportions. Relative figures in the latter case were a 6169-word training set obtained by scaling up 727 less common words, and a 531-word test set obtained by scaling up a disjoint 73 of the less common words. The number of (non-blank) characters in the two training sets, and in the two test sets, was the same. Results are reported to be "quite surprising" in that the network trained on the less common words (presumed to be more regular in their pronunciations) did not perform better than the network trained on the common words.

Additional light is thrown on this matter by Ainsworth and Pell (1989), who trained an MLP modelled on NETspeak on a 70,000-word dictionary divided into "regular" and "irregular" words according to whether pronunciation was correctly predicted by the Ainsworth (1973) rules or not. As one would expect, they found much better performance on the regular words than on the irregular words (asymptotic to 95.9% and 83.6% "best guess" phoneme scores on unseen words respectively). Thus, it seems that McCulloch *et al*'s assumption that the more common words have less regular pronunciations may be incorrect.

Unlike McCulloch *et al* whose concern was with the possible automatic creation of TTS systems, Seidenberg and McClelland (1989) studied the ability of a back-propagation network to simulate aspects of observed, human reading behaviour. Their network consisted of 400 input (orthographic) units, 200 hidden units and 460 output (phonological) units. Thus, there were 172,000 adjustable feedforward connections (excluding biases). There were also feedback connections (80,000) from the hidden to the orthographic units, but these do not appear to be essential to the operation of the model as implemented. Where NETtalk and NETspeak used a window of consecutive characters to provide context, Seidenberg and McClelland used input and output coding schemes which involve *triples* of consecutive ‘features’. A consequence of this is that any input word is encoded into a single input vector, irrespective of its length.

The input units each code 10 possible first characters (letters plus word-boundary marker), 10 possible second characters and 10 possible third characters, so that each specifies 1000 possible character triples. An input string turns a unit on if it contains a three-character substring which is one of the 1000 triples allocated to that unit. The allocation is done entirely at random except that the word-boundary marker cannot appear in the middle position. With this coding, the probability that two different input words would activate exactly the same set of units is effectively zero. Each of the output units represents a *single* triple of “phonetic features”. To keep the number of units tractable (460), however, Seidenberg and McClelland discarded every coding for which the first and third features referred to a different phonetic dimension. This aspect of the coding scheme, and others, have been much criticised in the literature (Pinker and Prince, 1988; Coltheart *et al*, 1993). In a later chapter, Plaut *et al* take up the issue of input/output representation in this sort of connectionist model.

This treatment of input and output strings – by allocating units to code the occurrence of particular sequences (triples) – has many important implications. It is apparent that no prior alignment of text and phonemes is required since both input and output strings are encoded as single vectors. However, while it is straightforward to determine the output coding for a particular phoneme string, as required in training, the converse operation of converting an output coding to a phoneme string is not possible. This makes it difficult if not impossible to determine whether the net’s output is correct or not. In our view, this inability to recover a phoneme sequence at the output is a virtually fatal flaw. It means that the model is incapable of simulating, for instance, the very basic human ability to name (say) the first phoneme of a printed word. How then did Seidenberg and McClelland overcome this profound objection?

In their model, a *phonological error score* is computed for any input as the sum of the squared differences between the obtained and desired output codings. By theorising that this score would correlate with naming latency in reading experiments, the predictions of the model could be checked against empirical observations. In fact, Seidenberg and McClelland achieve an impressive degree of success in this. They write (p. 540) that their “model simulates a broad range of empirical phenomena concerning the pronunciation of words and nonwords”. This error score also offered a possible way of deciding if the model’s output was ‘correct’. Seidenberg and McClelland computed all the output codings corresponding to the correct (‘target’) phoneme sequence plus all sequences differing by just one phoneme from the target. The output was then considered ‘correct’ if the sum of the squared differences corresponding to the target string was lower than that corresponding to any of the out-by-one-phoneme strings. As Coltheart *et al* (1993) point out, however, this procedure gives only a lower bound on the error rate of the model in pronouncing text strings.

Since psychological experimentation on reading traditionally favours the use of monosyllabic words, Seidenberg and McClelland used 2897 such words to train their net. Of this total, 2884 words were unique (e.g. the training set included (*wind*, /wɪnd/) and (*wind*, /waɪnd/) as separate 'words'). Training words were presented a number of times proportional to the logarithm of their frequency of occurrence in English. After training (on 150,000 word presentations), and using the somewhat suspect scoring method described above, only 77 (2.7%) of the words in the training set produced outputs which were deemed incorrect. Generalisation to unseen words was not tested systematically.

Before leaving the subject of back-propagation networks, it is worth pointing out a fundamental deficiency. Such nets, at least in their simple, feedforward form, are only suitable for mapping static input patterns to static output patterns. Because of the absence of feedback and/or memory, they are ill-suited to processing dynamic input sequences. Yet the input (and output) strings encountered in text-phoneme translation are inherently sequential. This necessitates essentially *ad hoc* fixes like the use of a sliding context window (in NETtalk and NETspeak) or the wholly-inappropriate input/output coding employed by Seidenberg and McClelland. A more principled way to proceed would be to employ dynamic, recurrent nets – as also suggested in the following chapters.

3.6 Analogy-based methods

The use of letter-to-sound rules in conjunction with a dictionary of pronunciations to some extent mirrors dual-route, psychological models of reading aloud (see above). As stated above, arguments for dual-route theory are based on the ability to pronounce pseudowords, latency difference effects between regular and exception words, and apparent double dissociation between the two routes in dyslexia. However, it has been variously argued that all these observations can be explained by a *single* route. One pervasive idea in the literature is that pseudowords are pronounced *by analogy* with lexical words that they resemble (Baron, 1977; Brooks, 1977; Glushko, 1979; 1981; Brown and Besner, 1987). Glushko, for instance, showed that "exception pseudowords" like *tave* take longer to read than "regular pseudowords" such as *taze*. Here, *taze* is considered as a "regular pseudoword" since all its orthographic 'neighbours' (*raze, gaze, maze* etc.) have the regular vowel pronunciation /eɪ/.) By contrast, *tave* is considered to be an "exception pseudoword" since it has the exception word (*have*, /hav/) as an orthographic neighbour. Thus, in the words of Glushko (1979), the "... assignment of phonology to non-words is open to lexical influence" – a finding which is at variance with the notion of two separate, independent routes to pronunciation. Instead of this:

> "... it appears that words and pseudowords are pronounced using similar kinds of orthographic and phonological knowledge: the pronuncia-

tion of words that share orthographic features with them, and specific spelling-to-sound rules for multiletter spelling patterns."

Thus, in place of *abstract* GPC rules in the dual-route model we have *specific* patterns of correspondence in the single-route, analogy model.

Pronunciation by analogy can be either *explicit* or *implicit*. The explicit form (e.g. Baron, 1977) is a conscious strategy of recalling a similar word and modifying its pronunciation, whereas in implicit analogy (e.g. Brooks, 1977) a pronunciation is derived from *generalised* phonographic knowledge about existing words. Implicit analogy has obvious commonalities with single-route, connectionist models (e.g. Sejnowski and Rosenberg, 1987; Seidenberg and McClelland, 1989). This commonality is pointed up by Glushko's arguments (1979, pp. 686–687; 1981, pp. 71–72) for the term *activation* in place of *analogy* since, for him, the process is naturally unconscious (i.e. implicit).

To test the computational feasibility of (explicit) analogy, Dedina and Nusbaum (1991) produced a prototype TTS system called *PRONOUNCE*. According to these authors "pronunciation-by-analogy may provide the same pronunciation ability as a set of spelling-to-sound rules without requiring an explicit theory of rule induction ... and may be relatively simple to automate." They identify the principal theoretical issue as "the degree to which orthographic consistency in the spelling patterns of words is related to phonographical consistency in pronouncing these words."

The lexical database of *PRONOUNCE* consists of a 20,000-word dictionary in which text and phonemes have been aligned. Dedina and Nusbaum acknowledge the crude nature of their alignment procedure, saying it "was carried out by a simple Lisp program that only uses knowledge about which phonemes are consonants and which are vowels." An incoming word is matched against orthographic entries in the lexicon by a process of registering the spelling patterns relative to one another and evaluating the number of contiguous, common letters for each registration index. Matched substrings, together with their phonemic mappings as stored in the lexical database, are used to build a pronunciation lattice which is then traversed to find a set of possible pronunciations for the input word. Pronunciations are rank-ordered, first by length of path through the lattice and, second, by the sum of the arc frequencies, reflecting the number of matched substrings that produced that arc. (The system has no knowledge of specific word frequencies.) When tested on the 70 pseudowords employed by Glushko in his (1979) study, *PRONOUNCE* exhibited an error rate of 9%; here, a 'correct' pronunciation is taken as one produced by any of Dedina and Nusbaum's 7 human subjects. By contrast, the well-known rule-based system DECtalk had an error rate of 3%. The authors interpret their results as showing "that pronunciation-by-analogy is computationally sufficient to generate reasonable pronunciations for short novel strings".

In explicit analogy, there is (apart from text-phoneme alignment) no prior training or inferencing phase. An input word is merely matched – as described above – with every orthographic entry in the lexicon and a pronunciation inferred from the matching substrings. This implies a good deal of computation to produce a pronunciation.

Subsequently, Sullivan and Damper (1992; 1993) extended this work in various ways. They employ an improved alignment procedure based on the Lawrence and Kaye (1986) algorithm. By pre-computing mappings and their statistics for use in the matching process, they have implemented a considerably more 'implicit' form of synthesis-by-analogy than did Dedina and Nusbaum. This pre-computation from the lexicon of possible mappings and their statistics amounts to a form of self-learning. They have also examined different ways of numerically ranking the candidate pronunciations. The analogy process is extended to the phonemic (in addition to the orthographic) domain. This latter extension has necessitated a reversion to some form of rules, in order that 'plausible' pronunciations for an unknown word can be generated and matched against lexical entries. The "flexible" grapheme-to-phoneme rules of Brown and Besner (1987) are used for this purpose, where "flexible" means context-independent. Since the intention is to produce a *set* of plausible pronunciations, there is no necessity to resolve conflicts to ensure that only a single rule is fired. (Of course, the use of left and right contexts is an important strategy for conflict resolution in traditional rule-based transcription.)

Stanfill (1987) describes MBRtalk, a self-learning text-phoneme conversion system based on so-called *memory-based reasoning* (MBR). In our terms, however, this can be viewed as an analogy-based technique. The basic principle of MBR is that "best-match recall from memory" can be regarded as a "primary inference mechanism". For every letter of every word in the dictionary of aligned orthographic-phonemic word pairs, a *frame* is created having 5 fields: the letter itself, the previous 4 letters, the succeeding 4 letters, the (single) phoneme aligned to that letter, and the stress assigned to it. Stanfill omits to say how many conflicting frames (having the same *letter*, *left-context* and *right-context* fields but different *phoneme* and *stress* fields), if any, ever arose. (Information on this point would give useful insight into the inherent difficulty of the transcription task.)

The complete set of frames is stored in memory for use during transcription. As with NETtalk and NETspeak, the assumption of a single-letter to single-phoneme alignment requires that a special character ("–") be used to denote a *null* phoneme, and that dipthongs etc. be treated as single phonemes. In the case of a (possibly novel) input word, there will one frame for each letter of the word but the *phoneme* and *stress* fields of these frames will be empty; it is the task of MBRtalk to fill them. This is done by comparing each frame of the test word with every frame in memory and retrieving the best-matching frame. The relevant contents of the retrieved memory frames are then transferred to the test frames to give a pronunciation.

This strategy of finding the best match to a lexically-specified pronunciation stored in memory leads us to contend that MBRtalk is using a form of synthesis-by-analogy. The frames are effectively 'analogy segments', i.e. fixed (9-character) substrings which, when matched, provide a single phoneme in the output pronunciation. Unlike *PRONOUNCE*, however, there is a significant pre-training phase consisting of extracting and storing in memory frames derived from the lexical database. Also, the simple mechanism of stepping the 9-character window through the input word a character at a time and concatenating output phonemes corresponding to the central letter (in the manner of NETtalk and NETspeak) avoids any need for a pronunciation lattice.

Clearly, the efficacy of MBRtalk depends critically upon having good similarity metrics. Dissimilarity between two frames is computed by assigning a (heuristically-chosen) penalty for each field in which they conflict. With a 1024-frame (approximately 200-word) test set, Stanfill found a best accuracy of 88% frames correct with a 132,072-frame (about 25,000-word) training dictionary. This result was obtained with a penalty function which depended solely on the contents of the conflicting fields of the frames of the test word. Obviously, it would be more principled to score penalties according to the (conflicting) contents of fields for both test and training data. Indeed, the necessity to use (non-optimal) heuristics for scoring similarity seems currently to be a major weakness of synthesis-by-analogy.

In subsequent work, Stanfill (1988) presents a rule-induction methodology within the framework of the MBR paradigm. The system, called *JOHNNY*, starts with "some knowledge of phonetic rules plus a phonetic lexicon". Unusually, there is no orthographic lexicon (or, at least, it is initially empty). As shown in Figure 4, the basic operations are rule application, "recognition", memorisation and induction. All of these are implemented using MBR.

There are 104 rewrite rules, using a 3-letter window as their left-hand side (LHS). The left and right contexts are a maximum of one character, which can be "don't care". An input word is broken into a series of such 3-letter windows which are each matched against the LHSs of the rule set in memory. Mismatches are penalised such that the best matches will be the most specific rules which are applicable. The input word is assumed to be present in the phonetic vocabulary. The rules are ambiguous in that this process produces a number of plausible pronunciations.

"Recognition" scores these candidate pronunciations against the phonetic vocabulary, again using MBR. Testing with a text of 1024 dictionary-derived pronunciations of words spontaneously uttered by a child in the first grade, the error rate at this stage (rules plus recognition) was 10%. For 8192 words randomly selected from a dictionary, however, the error rate was lower at 7%. The implication is that randomly-selected dictionary entries are more regular in their pronunciation than spontaneous utterances – at least, those of a child in first grade.

Having obtained this pronunciation, *JOHNNY* assumes it is correct and commits it to its 'learned' memory, using exactly the same 9-character frame representation and structure as MBRtalk. (Stanfill uses the terminology *frame* and *record* interchangeably in his 1987 paper; in the 1988 paper, he uses *record* exclusively but we prefer to retain *frame*.) This memorisation is seen as a process of acquiring spelling-pronunciation pairings in unsupervised fashion. While the unsupervised nature of

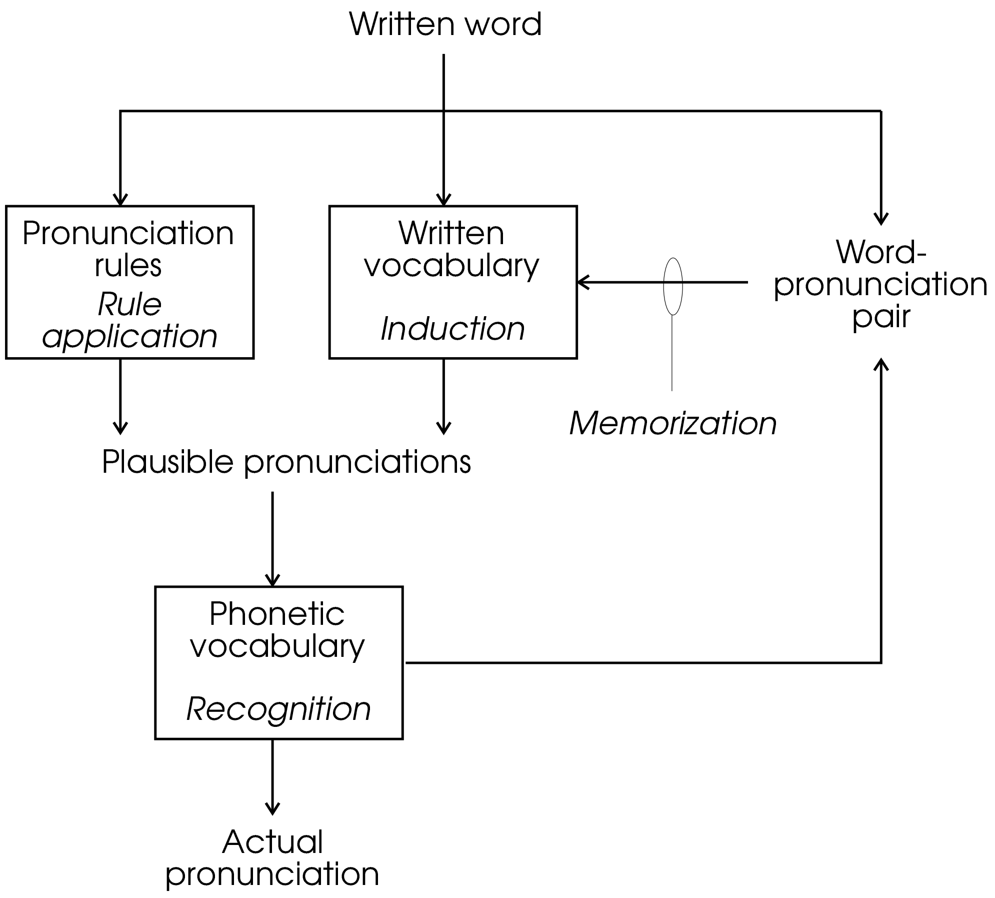

Figure 4: Schematic of *JOHNNY*, a memory-based model of the process of "learning to read". *After Stanfill (1988).*

the learning is one of the system's most intriguing aspects, it seems likely that the success of the learning is critically dependent upon the 104 initial rules and on using as input only those words present in the "phonetic vocabulary".

Stanfill is not explicit on the mechanism of rule induction. It appears, however, that the rules and what we have called the 'learned' memory are merely used in parallel to produce candidate pronunciations, as depicted in Fig. 4. This impression is strengthened by the fact that rules and learned memory have different structures, i.e. a 3-character window and 9-character frame respectively. It is not clear why it is necessary to memorise a "sufficient number of words" before using the 'learned' memory to supplement the rules in this way.

When the rules are supplemented in this way, the error rates on the above test materials (after 2 passes) fall from 10% to approximately 7% for the 'first grade' words, and from 7% to approximately 3% for the randomly-selected dictionary words. *JOHNNY* also displays an impressive lack of sensitivity to the initial rule set – as demonstrated by, for example, deleting all multi-letter rules and showing that the system is still capable of learned improvement. Finally, and not unexpectedly, *JOHNNY* is capable of further improvement when the learning is supervised (i.e. the system is told when it makes a mistake, and might also be given the correct pronunciation) rather than unsupervised.

Recently, Van den Bosch and Daelemans (1993) have described a very similar approach to memory-based reasoning that they describe as "a link between straightforward lexical lookup and similarity-based reasoning". The method takes a pronouncing dictionary as the training set but "solves the problem of lacking generalisation power and efficiency by *compressing* it into a text-to-speech *lookup table*". Applied to Dutch, they say: "The most surprising result of our research is that the simplest method (based on tables and defaults) yields the best generalisation results, suggesting that previous knowledge-based approaches were overkill".

3.7 Syntactic neural networks

Statistical translation models rely on associating 'units' in one language, or domain, with 'units' in another. Lucas and Damper (1992) describe a text-phoneme conversion model based such statistical ideas and having a direct connectionist implementation in terms of so-called syntactic neural networks. They attempt to translate between commonly-occurring substrings in the two domains. The assumption is that common substrings represent potentially useful abstractions, as in the case of *th*.

The training process consists of three passes through the dictionary of orthographic-phonemic word pairs. First, the m most probable substrings (n-grams) in each domain are enumerated. The starting point is the alphabets of atomic symbols, Σ_o and Σ_p, where the subscripts denote orthography and phonemics respectively. These atomic symbols are thought of as the terminals of a simple grammar for the generation of the observed words. Each word encountered (e.g. *the*) is 'exploded' into its constituent substrings (i.e. *t, h, e, th, he* and *the*). The cumulative count for each such substring is then incremented before considering the next word. Thus, a set (initially empty) of 'non-terminal' n-grams is built up, to form new alphabets N_o and N_p such that $A_o = \Sigma o \cup N_o$, an alphabet of orthographic symbols, and likewise an alphabet of phonemic symbols A_p.

At termination of the first pass, substrings are sorted into frequency order and the first m (which include the atomic symbols) of each taken as the translation alphabets, i.e. $|A_o| = |A_p| = m$. The corresponding inferred grammar is trivial in the sense that each non-terminal can only rewrite to a single string of two 'lower-order' symbols (i.e. productions are of the form $(th) \rightarrow (t)(h)$).

Following this initial pass, the second phase of learning calculates the bigram statistics of the symbols in A_o and A_p in their respective domains. For the example case of *the*, this is effectively considered to be the four distinct words $(t)(h)(e)$, $(th)(e)$, $(t)(he)$ and (the) in evaluating the bigram statistics. These are importantly different to 'ordinary' bigram statistics, since many of the (non-terminal) symbols correspond to several atomic characters. In effect, we have an adaptive n-gram description of strings in each domain – in that the value of n varies so as to give the most compact representation for any given size of alphabet. The bigram statistics give the within-domain transition probabilities of going from one symbol to another.

Until this point, the orthographic and phonemic domains have been considered separately. At the third pass, however, words are first considered as orthographic-phonemic pairs. This phase estimates the cross-domain (translation) probabilities $P(o \rightarrow p|o)$, i.e. given $o \in A_o$, the probability that o rewrites to a particular $p \in A_p$. This is done as follows. The possible segmentations of each word pair in the dictionary with respect to the alphabets A_o and A_p are identified, and used to build a pronunciation lattice. This is depicted in Figure 5 for the example word (*zoom*, /zuːm/). (Note the treatment of the vowel lengthening symbol /ː/ as an atomic symbol although it only ever occurs in conjunction with a vowel. The system was left to infer this for itself.) Although not shown in the figure, transition probabilities are attached to each arc. A possible alignment then corresponds to a pair of paths between word start and end, one in the orthographic domain and the other in the phonemic domain, and having an equal number of arcs. The set of such possible alignments, $\{A_i\}$, is enumerated using a path algebra. For the example word, these are:

A_1	:	((*z*, /z/) (*o*, /u/) (*o*, /ː/) (*m*, /m/) 1.6×10^{-8})
A_2	:	((*z*, /z/) (*oo*, /u/) (*m*, /ːm/) 8.9×10^{-7})
A_3	:	((*z*, /z/) (*oo*, /uː/) (*m*, /m/) 1.7×10^{-6})

where the numbers represent the product of all arc (transition) probabilities on the respective o and p paths.

At the end of the pass, these figures are used to compute mapping probabilities for all the $(o \rightarrow p)$ correspondences seen. This is done by, for a particular i and j, simply counting the number of occurrences of $o_i \rightarrow p_j$ correspondences in the training data, and normalising by the total number of correspondences involving o_i.

Although associations are (in principle) allowed between o and p substrings in any position within a word, the method enumerates contiguous paths only. Thus, non-contiguous markings (as with the final *e* of (*make*, /meɪk/) indicating the dipthongisation of the medial vowel) cannot be dealt with at all directly. In effect, the system has to learn that *ke* sometimes functions as an orthographic unit which associates with phoneme /k/ following /eɪ/. A related point is that there is no simple way to allow symbols in the source domain (letters here) to rewrite to *null* in the target domain.

As the enumeration of possible alignments is an intrinsic part of the learning process, the necessity for aligned training data is avoided. To this extent, the learning is unsupervised in that no target output is specified for each input symbol.

There are many ways the described model and its estimated probabilities could be used in a translation system. For instance, given a test word, its (orthographic)

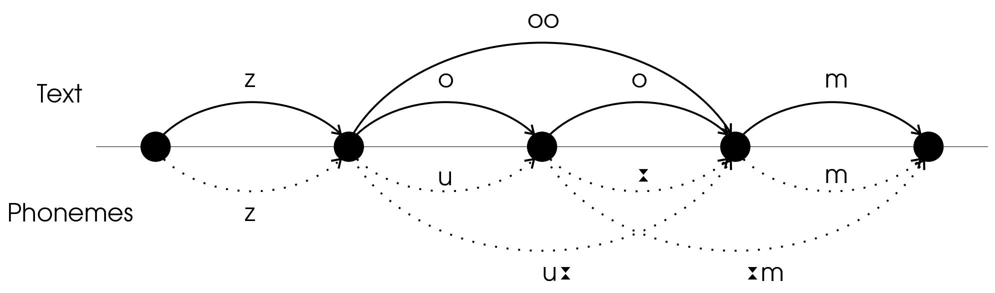

Figure 5: Pronunciation lattice for the word (*zoom*, /zuːm/), given the non-terminals inferred by the system. A possible alignment corresponds to a pair of paths between word start and end, one in the orthographic domain and the other in the phonemic domain, and having an equal number of arcs. *After Lucas and Damper (1992).*

segmentations could be determined and the translation (sequence of phoneme units) selected having the highest joint probability of the *o* path and associated rewrites for the arcs on that path. Another possibility would be to consider the *p* path probability additionally. However this is done, treating the probabilities as weights in a neural network allows a connectionist implementation to be conceived, as detailed in Lucas and Damper (1992). A similar connectionist model has recently been reported by Smith (1993) who employs an argument based on eye movements during reading to support the idea of sub-word units corresponding to Lucas and Damper's non-terminal n-grams.

While the performance obtained is not especially impressive (performance asymptotic to about 67% phonemes correct on largely monosyllabic words as test and training set increase in size), the authors point to "the system's immaturity" and describe their methods as "theoretically interesting from the point of view of … text-phonetics correspondence in general".

4 Summary

A critical review of self-learning approaches to text-phoneme conversion has been presented. To our knowledge, this is the first such review to appear. Work on this topic has been approached from two different perspectives: the automatic production of TTS (and possibly speech-to-text) systems, reducing the amount of manual work which must be done in implementing a new language as well as giving a potential means of improving existing rule-based systems, and the modelling of human reading processes – especially the acquisition of the skill of reading aloud. The chapter is intended to provide a framework and introduction for the following contributions which focus on connectionist models of reading.

The standard approach to text-phoneme conversion (in the absence of a dictionary containing all words ever encountered) in TTS systems uses a set of context-dependent translation rules. The operation of such rules has been described, and automatic-discovery techniques for inferring CDT rule-sets reviewed. However, the main focus of this chapter is on self-learning, connectionist approaches. The rule-based approaches are included for completeness and as a basis of comparison.

Most commonly, connectionist models use the popular error back-propagation learning – as in NETtalk, NETspeak and the work of Seidenberg and McClelland. While the later model codes inputs and outputs in terms of a fixed set of features for any given word, NETtalk and NETspeak employ a sliding window which passes over the input character-by-character. Accordingly, they confront a 'bootstrapping' problem in that they require text pre-aligned with corresponding phonemes for training. This implies prior knowledge of the correspondence between text and phonemes that we are actually trying to infer. Hence, interest centres on connectionist models which do not require pre-alignment, such as that of Bullinaria to be described shortly in this volume. Another problem is that, at least in their simple feedforward form, back-propagation networks are really only suitable for processing static patterns. The use of dynamic, recurrent nets appears highly appropriate, yet has not so far (to my knowledge) been reported in the literature.

Hidden Markov models have been popular in speech recognition and there has been some interest in exploiting the trainable nature of HMMs in text-phoneme conversion. Connectionist implementations of HMMs have also been described in the literature.

Recently synthesis-by-analogy has received renewed attention as a computational TTS model and as a single-route alternative to dual-route theory. In its explicit form, an unknown word is compared with lexical entries and a pronunciation assembled from matching segments. In its implicit form, generalised knowledge concerning text-phoneme correspondences is pre-compiled and used in translation. Implicit analogy is close in spirit to connectionist approaches. Of course, much depends upon how matching is performed in the explicit case, and knowledge captured in the implicit case. Because of this, a number of approaches can be classified within the analogy framework. Examples are Stanfill's MBRtalk (founded on memory-based reasoning) and Van den Bosch and Daelemans' compressed table lookup method.

Ideas of statistical association and translation between features (letter or phoneme units) in the two different domains have been used by Lucas and Damper, and by Smith, to produce connectionist models of reading aloud that avoid the need for pre-alignment. As yet, these perform relatively poorly but they do represent an interesting direction for future research.

References

ABERCROMBIE, D. (1981) “Extending the Roman alphabet: some orthographic experiments of the past four centuries”, in *Towards a History of Phonetics*, edited by R.E. Asher and E. Henderson (Edinburgh University Press, Edinburgh, UK), pp. 207–224.

AINSWORTH, W.A. (1973) “A system for converting English text into speech”, *IEEE Trans. Audio Electroacoust.*, **AU-21**, 288–290.

AINSWORTH, W.A. AND PELL, B. (1989) “Connectionist architectures for a text-to-speech system”, *Proc. Eurospeech ’89*, Paris, 125–128.

ALLEN, J. (1976) “Synthesis of speech from unrestricted text”, *Proc. IEEE*, **64**, 422–433.

ALLEN, J., HUNNICUTT, M.S. AND KLATT, D.H. (1987) *From Text to Speech: the MITalk System* (Cambridge University Press, Cambridge, UK).

BARON, J. (1977) “Mechanisms for pronouncing printed words: use and acquisition”, in *Basic Processes in Reading: Perception and Comprehension*, edited by D. LaBerge and S. Samuels (Lawrence Erlbaum, Hillsdale, NJ), pp. 175–216.

BAUM, L.E. (1972) “An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes”, *Inequalities*, **3**, 1–8.

BRIDLE, J.S. (1990) “Alpha nets: a recurrent neural network architecture with a hidden Markov model interpretation”, *Speech Communication*, **9**, 83–92.

BROOKS, L. (1977) “Non-analytic correspondences and pattern in word pronunciation”, in *Attention and Performance VII* edited by J. Renquin (Lawrence Erlbaum, Hillsdale, NJ), pp. 163–177.

BROWN, P. AND BESNER, D. (1987) “The assembly of phonology in oral reading: a new model”, in *Attention and Performance XII: The Psychology of Reading*, edited by M. Coltheart (Lawrence Erlbaum Assoc., London), pp. 471–489.

CARLSON, R. AND GRANSTRÖM, B. (1975) “A phonetically-oriented programming language for rule description of speech”, in *Speech Communication, Vol. 2*, edited by C.G.M. Fant (Almqvist and Wiksell, Uppsala, Sweden), pp. 245–253.

CARLSON, R., GRANSTRÖM, B. AND HUNNICUTT, S. (1990) “Multilingual text-to-speech development and applications”, in *Advances in Speech, Hearing and Language Processing, Vol. 1*, edited by W.A. Ainsworth (JAI Press, London), pp. 269–296.

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

Church, K.W. (1985) "Stress assignment in letter-to-sound rules for phonological rules for speech synthesis", *Proc. 23rd Meeting Assoc. Comp. Ling.*, Chicago, IL, 246–253.

Coltheart, M. (1978) "Lexical access in simple reading tasks", in *Strategies of Information Processing*, edited by G. Underwood (Academic, London), pp. 151–216.

Coltheart, M., Curtis, B., Atkins, P. and Haller, M. (1993) "Models of reading aloud: dual-route and parallel-distributed-processing approaches", *Psychological Review*, **100**, 589–608.

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

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

Forney, G.D.Jr. (1973) "The Viterbi algorithm", *Proc. IEEE*, **61**, 268–278.

Forster, K.I. and Chambers, S.M. (1973) "Lexical access and naming time", *J. Verbal Learning and Verbal Behavior*, **12**, 627–635.

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

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

Hertz, S.R., Kadin, J. and Karplus, K.J. (1985) "The Delta rule development system for speech synthesis from text", *Proc. IEEE*, **73**, 1589–1601.

Humphreys, G.W. and Evett, L.J. (1985) "Are there independent lexical and nonlexical routes in word processing? An evaluation of the dual-route theory of reading", *The Behavioral and Brain Sciences*, **8**, 689–740.

Hunnicutt, S. (1976) "Phonological rules for a text-to-speech system", *Am. J. Comp. Ling.*, **Microfiche 57**, 1–72.

Klatt, D.H. (1987) "Review of text-to-speech conversion for English", *J. Acoust. Soc. Am.*, **82**, 737–793.

Klatt, D.H. and Shipman, D.W. (1982) “Letter-to-phoneme rules: a semi-automatic discovery procedure”, *J. Acoust. Soc. Am., Suppl. 1*, **72**, S48.

Lawrence, S.G.C. and Kaye, G. (1986) “Alignment of phonemes with their corresponding orthography”, *Computer Speech and Language*, **1**, 153–165.

Lucas, S.M. and Damper, R.I. (1992) “Syntactic neural networks for bi-directional text-phonetics translation”, in *Talking Machines: Theories, Models and Applications*, edited by G. Bailly and C. Benoît (Elsevier/North-Holland, Amsterdam), pp. 127–141.

Lucassen, J.M. and Mercer, R.L. (1984) “An information theoretic approach to the automatic determination of phonemic baseforms”, *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP ’84)*, San Diego, CA, 42.5.1–42.5.4.

Luk, R.W.P. and R.I. Damper (1991) “Stochastic transduction for English text-to-phoneme conversion”, *Proc. Eurospeech ’91, 2nd European Conf. Speech Communication and Technology, Vol. 2*, Genova, Italy, pp. 779–782.

Luk, R.W.P. and Damper, R.I. (1994) “A review of stochastic transduction”, *Proc. Joint European Speech Communication (ESCA)/IEEE Workshop on Speech Synthesis*, Lake Mohonk, New Paltz, NY, pp. 248–251.

Nádas, A. (1994) “Some relationships between ANNs and HMMs”, in *Artificial Neural Networks for Speech and Vision*, edited by R.J. Mammone (Chapman and Hall, London), pp. 240–267.

McCulloch, N., Bedworth, M. and Bridle, J. (1987) “NETspeak – a re-implementation of NETtalk”, *Computer Speech and Language*, **2**, 289–301.

Oakey, S. and Cawthorne, R.C. (1981) “Inductive learning of pronunciation rules by hypothesis testing and correction”, *Proc. Int. Joint Conf. Artificial Intelligence, IJCAI-81*, Vancouver, Canada, 109–114.

Oxford University Press (1989) *Oxford Advanced Learner’s Dictionary of Current English, 3rd Edition, Electronic Handbook* (Oxford University Press, Oxford, UK).

Parfitt, S.H. and Sharman, R.A. (1991) “A bidirectional model of English pronunciation”, *Proc. Eurospeech ’91, 2nd European Conf. Speech Communication and Technology, Vol. 2*, Genova, Italy, 800–804.

Pinker, S. and Prince, A. (1988) “On language and connectionism: analysis of a parallel distributed processing model of language acquisition”, *Cognition*, **28**, 73–193.

Quinlan, J.R. (1979) "Discovering rules from large collections of examples: a case study", in *Expert Systems in the Micro Electronic Age*, edited by D. Mitchie (Edinburgh University Press, Edinburgh, UK), pp. 168–201.

Quinlan, J.R. (1990) "Decision trees and decisionmaking", *IEEE Trans. Systems, Man and Cybernetics*, **SMC-20**, 339–346.

Rahim, M.Z. (1994) "A self-learning neural tree network for phone recognition", in *Artificial Neural Networks for Speech and Vision*, edited by R.J. Mammone (Chapman and Hall, London), pp. 227–239.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) "Learning internal representations by error propagation", in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, edited by D.E. Rumelhart and J.L. McClelland (Bradford Books/MIT Press, Cambridge, MA), pp. 318–362.

Scragg, D.G. (1975) *A History of English Spelling* (Manchester University Press, Manchester, UK).

Segre, A.M., Sherwood, B.A. and Dickerson, W.B. (1983) "An expert system for the production of phoneme strings from unmarked English text using machine-induced rules", *Proc. 1st Conf. European Chapter Assoc. Comp. Ling.*, Pisa, Italy, 35–42.

Seidenberg, M.S. and McClelland, J.L. (1989) "A distributed, developmental model of word recognition and naming", *Psychological Review*, **96**, 447–452.

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

Smith, J (1993) "Using unsupervised feature detectors in a model of reading aloud", *Irish Journal of Psychology*, **14**, 397–409.

Stanfill, C.W. (1987) "Memory-based reasoning applied to English pronunciation", *Proc. 6th National Conf. Artificial Intelligence, AAAI87*, Seattle, WA, 577–581.

Stanfill, C.W. (1988) "Learning to read: a memory-based model", *Proc. Case-Based Reasoning Workshop*, Clearwater Beach, FL, 406–413.

Stromberg, J.-E., Zrida, J. and Isaksson, A. (1991) "Neural trees – using neural trees in a tree classifier structure", *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP '91)*, Toronto, Canada, Vol. 1, 137–140.

SULLIVAN, K.P.H. AND DAMPER, R.I. (1992) "Novel-word pronunciation within a text-to-speech system", in *Talking Machines: Theories, Models and Applications*, edited by G. Bailly and C. Benoît (Elsevier/North-Holland, Amsterdam), pp. 183–195.

SULLIVAN, K.P.H. AND DAMPER, R.I. (1993) "Novel-word pronunciation: a cross-language study", *Speech Communication*, **13**, 441–452.

VAN COILE, B. (1990) "Inductive learning of grapheme-to-phoneme rules", *Proc. Int. Conf. Spoken Lang. Process. (ICSLP '90), Vol. 2*, Kobe, Japan, 765–768.

VAN DEN BOSCH, A. AND DAELEMANS, W. (1993) "Data-oriented methods for grapheme-to-phoneme conversion", *Proc. 6th Conf. European Chapter Assoc. Comp. Ling.*, Utrecht, **in press**.

VAN LEEUWEN, H.C. (1989) "A development tool for linguistic rules", *Computer Speech and Language*, **3**, 83–104.

VENEZKY, R.L. (1965) *A Study of English Spelling-to-Sound Correspondences on Historical Principles* (Ann Arbor Press, Ann Arbor, MI).

VENEZKY, R.L. (1970) *The Structure of English Orthography* (Mouton, The Hague, Netherlands).

VITERBI, A.J. (1967) "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", *IEEE Trans. Information Theory*, **IT-13**, 260–269.

WIJK, A. (1966) *Rules of Pronunciation of the English Language* (Oxford University Press, Oxford, UK).

WOLFF, J.G. (1984) "Inductive learning of spelling-to-phoneme rules", *Technical Report 128*, IBM UK Scientific Centre, Winchester, UK.