INTERNATIONAL JOURNAL OF SPEECH TECHNOLOGY 8, 149–162, 2005

Aligning Text and Phonemes for Speech Technology Applications Using an EM-Like Algorithm

R.I. DAMPER*
Image, Speech and Intelligent Systems (ISIS) Research Group, School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK
rid@ecs.soton.ac.uk

Y. MARCHAND
Institute for Biodiagnostics (Atlantic), National Research Council Canada, Neuroimaging Research Laboratory, 1796 Summer Street, Suite 3900, Halifax, Nova Scotia, Canada B3H 3A7

J.-D. S. MARSTERS AND A.I. BAZIN
Image, Speech and Intelligent Systems (ISIS) Research Group, School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK

Abstract. A common requirement in speech technology is to align two different symbolic representations of the same linguistic 'message'. For instance, we often need to align letters of words listed in a dictionary with the corresponding phonemes specifying their pronunciation. As dictionaries become ever bigger, manual alignment becomes less and less tenable yet automatic alignment is a hard problem for a language like English. In this paper, we describe the use of a form of the expectation-maximization (EM) algorithm to learn alignments of English text and phonemes, starting from a variety of initializations. We use the British English Example Pronunciation (BEEP) dictionary of almost 200,000 words in this work. The quality of alignment is difficult to determine quantitatively since no 'gold standard' correct alignment exists. We evaluate the success of our algorithm indirectly from the performance of a pronunciation by analogy system using the aligned dictionary data as a knowledge base for inferring pronunciations. We find excellent performance—the best so far reported in the literature. There is very little dependence on the start point for alignment, indicating that the EM search space is strongly convex. Since the aligned BEEP dictionary is a potentially valuable resource, it is made freely available for research use.

Keywords: text-to-speech synthesis, string alignment, dynamic programming, EM algorithm, pronunciation by analogy

1. Introduction

The requirement commonly arises in speech technology and natural language processing to align two linear, symbolic representations of the same linguistic entity. One important example, which forms the focus of this paper, is the alignment of the textual (orthographic or spelling) and phonemic (pronunciation) representations of isolated words (of English, in this work). The necessity to align text and phonemes arises in, for instance, inferring the complete form of spelling-pronunciation word pairs from elliptical entries in a dictionary (Lawrence and Kaye, 1986) and adding new entries to the pronunciation dictionary that provides a mapping between sub-word models and language models in automatic speech recognition (Knill and Young,

*To whom all correspondence should be addressed.

1997, p. 48). But as (Jansche, 2001) writes: “The problem of finding a good alignment has not received its due attention in the literature”.

Two examples from the domain of text-to-speech (TTS) synthesis suffice to motivate the search for powerful automatic alignment techniques.

1. In (supervised) training of neural networks to perform spelling-to-sound conversion, as in the well-known NETtalk and NETspeak of Sejnowski and Rosenberg (1987) and McCulloch et al. (1987) respectively, it is necessary to associate each letter of an input word with a target output phoneme. In both works, alignment was done manually, but this is time-consuming, error-prone, and limits the size of datasets that can be used for training. As speech synthesis becomes ever more data-driven (Damper, 2001) using ever larger dictionaries and corpora (Young and Bloothooft, 1997), so manual alignment becomes less and less tenable and the need for automatic alignment methods increases.
2. Increasingly in recent years, an approach known as *pronunciation by analogy* (PbA) has been used in TTS synthesis to derive pronunciations for unknown words, i.e., those not listed in the system dictionary (Dedina and Nusbaum, 1991; Sullivan and Damper, 1993; Pirrelli and Federici, 1994; Pirrelli and Federici, 1995; Federici et al., 1995; Damper and Eastmond, 1996; Yvon, 1996a; Yvon, 1996b; Damper and Eastmond, 1997; Bagshaw, 1998; Damper et al., 1999; Pirrelli and Yvon, 1999; Marchand and Damper, 2000; Sullivan, 2001). PbA assembles pronunciations for such (unknown) words from partial matches to the (known) words listed in the dictionary—a process that requires each letter of every word in the dictionary to be aligned with a corresponding phoneme in contiguous, one-to-one fashion.

However, automatic alignment is a difficult problem. Much of the difficulty arises because of the lack of regularity (‘consistency’ and ‘transparency’) in the English writing system. By ‘consistency’, we mean that the same letter always corresponds to the same phoneme. In fact, English is notorious for the lack of consistency in its spelling-to-sound correspondence (Venezky, 1965; Carney, 1994) at the level of single letters. For instance, the letter *c* is pronounced /s/ in *cider* but /k/ in *cat*. On the other hand, the /k/ sound of *kitten* is written with a letter *k*. By ‘transparency’, we mean that a single letter corresponds to a single phoneme (Henderson, 1984, p. 17) and vice versa.

The lack of consistency in English orthography is problematic for alignment since any given letter can potentially align with (i.e., correspond to) many different phonemes. To illustrate the problems that arise from lack of transparency, consider the word (*quay*, /ki/), for which a reasonable alignment might be:

q	u	a	y
|	|	|	|
k	_	i	_

This word is not unusual for English in having fewer phonemes than letters, necessitating the insertion of ‘null phonemes’ in the transcription if a one-to-one mapping is to be maintained. Such null symbols are entirely ‘artificial’ in that they play no role in specifying the pronunciation; their only purpose is to maintain the one-to-one correspondence between letters and phonemes. Yet it is not clear precisely where the null letters should be placed, since the following is also a reasonable alignment:

q	u	a	y
|	|	|	|
_	k	_	i

This example illustrates a key aspect of the lack of transparency in that letter combinations frequently correspond to a single phoneme—a form of context dependency. Such letter combinations have been called “functional spelling units” (Venezky, 1970; Coltheart, 1984). Examples of functional spelling units are *th* → /ð/ as in *that*, *ch* → /tʃ/ as in *church*, and *qu* → /k/ as in this example of *quay*. Unfortunately, any of the letters of the functional spelling unit could plausibly align with the corresponding phoneme, with the others corresponding to nulls, leading to a degree of indeterminacy.

More rarely, there are fewer letters than phonemes in a word of English. Examples are (*six*, /sɪks/) and (*sex*, /sɛks/) in which the single letter *x* maps to the two phonemes /ks/, so that ‘null letters’ may have to be introduced to maintain a one-to-one mapping. Sejnowski and Rosenberg (1987) actually invented ‘new’ phonemes (/K/, /X/ and /#/) in NETtalk to avoid introducing null letters. As with null phonemes, the problem arises as to exactly where the nulls should be

P1: LOP
INTERNATIONAL JOURNAL OF SPEECH TECHNOLOGY SJNW281-06-DAMPER June 1, 2005 5:58

placed. Worse yet, both problems—null letters and null phonemes—can occur in the same word, as in the case of (*axe*, /aks/) for which a reasonable alignment is:

a	x	_	e
|	|	|	|
a	k	s	_

So the simple-minded presumption that the same number of letters and of phonemes implies a one-to-one mapping is mistaken in this case.

These examples illustrate that there is no canonically correct alignment of text and phonemes in every case, nor should we expect this, since the process is essentially a computational convenience lacking any sound linguistic or theoretical basis. The alignment problem is especially severe for languages like English and French whose writing systems are 'deep', i.e., they display a complex relation between spelling and sound lacking consistency and transparency, unlike the 'shallow' orthographies of Finnish or Serbian for example, where the correspondence is mostly if not entirely consistent and transparent (Coltheart, 1978; Liberman et al., 1980; Katz and Feldman, 1981; Turvey et al., 1984; Sampson, 1985). Indeed, (Abercrombie, 1981, p. 209) describes the English spelling-to-sound system as "… one of the least successful applications of the Roman alphabet."

As one last illustration of the complexities of spelling-sound correspondence in English, consider the word (*made*, /meɪd/):

m	a	d	e
|	|	|	|
m	eɪ	d	_

Here, the final *e* aligns with a null phoneme, yet it does not seem natural to view *de* as a functional spelling unit in this case. Removing the *e* yields the word (*mad*, /mad/), so that it acts as a 'marking' (Venezky, 1970), signifying that the preceding vowel is lengthened or dipthongized: /a/ becomes /eɪ/. This contrasts with the final *e* of *axe*, which has no such marking effect, further illustrating the inconsistent and partly-arbitrary nature of the English spelling system. Markings in English, whereby a final letter affects the sound of a medial vowel letter, can be very long range, as in the well-known example word pairs *photograph*/*photography* and *telegraph*/*telegraphy* (Chomsky and Halle, 1968). They can be seen as an interaction of the lack of consistency and transparency, both of which—as we have seen—complicate the process of alignment.

Given these difficulties, it is clear that the automatic alignment of text and phonemes is not a straightforward matter. In the remainder of this paper, we develop an approach to alignment based on ideas originally found in Luk and Damper (1991, 1992, 1993, 1996), but using much-improved algorithms. Although imperfect, our earlier methods have in fact been used by other authors (e.g., Parfitt and Sharman, 1991;Jansche, 2001), reflecting the widespread need for a good alignment algorithm.

2. Alignment by Dynamic Programming

Dynamic programming (Bellman, 1957; Kruskal, 1983) offers a simple and powerful way to align text and phonemes on the assumption that we have some knowledge of the probability of a particular letter mapping to a particular phoneme. In this work, knowledge about letter-phoneme mappings will be compiled in an 'association' matrix, **A**, of dimension $L \times P$, where L is the size of the letter inventory (i.e., 26) and P is the size of the phoneme inventory (which is 44 here). The dynamic programming (DP) principle asserts that the global solution to a path-finding problem can be found by a sequence of locally-optimal steps; in other words, no local non-optimality can contribute to a globally-optimal solution. This principle is well-known and widely-used in computational linguistics and speech technology, forming for instance the basis of the CYK parsing algorithm (Hopcroft et al., 2001, pp. 298–301) and the Viterbi algorithm (Viterbi, 1967; Forney, 1973; Neuhoff, 1975), used in various guises in speech recognition, speech synthesis, and text processing.

The process of aligning text and phonemes for a specific word can be cast as a path-finding problem by building a table, or **B** matrix, indexed by the letters of the word's spelling and the phonemes of its pronunciation. This is illustrated for the word (*phase*, /feɪz/) in Fig. 1(a). The entries in this matrix are to be interpreted as degrees of 'association' between each letter and each phoneme. The procedure for inferring these entries is detailed in later sections. (The values seen here are taken from one iteration of an actual run of our algorithm.) Note that we have added word delimiters (# and $ for letter and phoneme domains respectively), with an association of 0 for (#, \$). This is done to allow the DP algorithm to align the leading letter or phoneme of a word with a null; otherwise the first letter would always align with the first phoneme. The 'best' alignment of letters and phonemes is then defined by the path from the top-left entry of the matrix to the bottom-right that maximizes the accumulation of association values along this path.

To find this best alignment, we introduce two new matrices **C** and **D**. Matrix **C** is a table of accumulated associations, such that each entry is the maximum accumulated association up to that point in the table (i.e., up to that point in the alignment). Matrix **D** holds pointers indicating the precursor cell from which the DP algorithm moved to each cell. The **C** and **D** matrices are filled left-to-right, top-to-bottom using some appropriate form of simple recursive maximization equation. At the end of the process, the **C** matrix holds the maximum accumulated association for the complete word in its bottom right cell, and the best alignment can be found by tracing pointers back from the bottom right cell of the **D** matrix.

In this work, we have used the implementation of DP due to Needleman and Wunsch (1970), since it is simple, well-known and performed very satisfactorily in preliminary, exploratory investigations. The specific form of the recursive maximization equation for a given word w is:

$$C_{i,j} = \max \left\{ \begin{array}{l} C_{i-1,j-1} + B_{i,j}, \\ C_{i-1,j} - \delta, \\ C_{i,j-1} - \delta \end{array} \right\} \qquad \begin{array}{l} 1 \le i \le |l_w| \\ 1 \le j \le |p_w| \end{array} \tag{1}$$

where $|l_w|$ and $|p_w|$ are the lengths of word w in terms of letters and phonemes (including delimiters) respectively, and δ is some suitably chosen penalty term, which here is set to 0.

Figure 1(b) shows the **C** and **D** matrices found for the word (*phase*, /feɪz/) with the associations tabulated in Fig. 1(a). For ease of illustration, the two matrices are shown superimposed. If the maximization chose the $C_{i-1,j-1} + B_{i,j}$ argument, corresponding to a diagonal move in the **B** and **C** matrices, the entry in the **D** matrix is "↘". If the maximization chose the $C_{i-1,j}$ argument, corresponding to a vertical move in the **B** and **C** matrices, the entry in the **D** matrix is "↓", corresponding to alignment of a letter with a null phoneme. If the maximization chose the $C_{i,j-1}$ argument, corresponding to a horizontal move in the **B** and **C** matrices, the entry in the **D** matrix is "→", corresponding to alignment of a phoneme with a null letter. The "ϵ" in the top left cell indicates the start for the DP alignment from which no back-tracing is possible. The maximal association (or DP score) for the word is align(*phase*) = 71446. By tracing pointers back from the bottom right entry, the alignment is found as:

p	h	a	s	e
|	|	|	|	|
_	f	eɪ	z	_

Note that the dynamic programming handles context dependency (e.g., letter group *ph* acts here as a functional spelling unit) in an implicit manner, since at each step of the maximization, Eq. (1), we consider moves from the three possible precursors (cells $(i-1, j-1)$, $(i-1, j)$, and $(i, j-1)$) of cell (i, j). At the same time, the very strong $a \rightarrow$ /eɪ/ and $s \rightarrow$ /z/ associations of 23098 and 45788 respectively in Fig. 1 act as 'anchors' for the DP alignment.

It only remains to find the **A** matrix and thereafter we can align any word in the dictionary. This is done

	\$	f	eɪ	z	\$
#	0	0	0	0	0
p	0	9	0	0	0
h	0	2580	27	35	0
a	0	42	23098	937	0
s	0	79	3	45788	0
e	0	947	1732	2641	0
#	0	0	0	0	0

(a)

	\$	f	eɪ	z	\$
#	0, ϵ	0, →	0, →	0, →	0, →
p	0, ↓	9, ↘	9, →	9, →	9, →
h	0, ↓	2580, ↘	2580, →	2580, →	2580, →
a	0, ↓	2580, ↓	25678, ↘	25678, →	25678, →
s	0, ↓	2580, ↓	25678, ↓	71446, ↘	71446, →
e	0, ↓	2580, ↓	25678, ↓	71446, ↓	71446, ↘
#	0, ↓	2580, ↓	25678, ↓	71446, ↓	71446, ↘

(b)

Figure 1. (a) Example matrix of letter-phoneme associations (**B** matrix) for the word (*phase*, /feɪz/). The word is delimited by # and \$ in the letter and phoneme domains respectively. See text for explanation of entries. (b) Table of cumulative associations found by dynamic programming, together with the production or 'move' from the precursor cell that maximizes this value. This table can be viewed as a superposition of **C** and **D** matrices (see text).

using a form of the EM algorithm, which is the subject of the next section.

3. Estimating Associations with the EM Algorithm

The expectation-maximum (EM) algorithm is an iterative approach to the solution of maximum-likelihood estimation problems when there are data missing from the set of observations and/or the likelihood function cannot be easily differentiated to find its maxima. Although the basic idea had appeared in the literature previously (e.g., Hartley, 1958; Baum, 1972), the term "EM algorithm" was coined by Dempster et al. (1977). A useful introduction is provided by Moon (1996); an excellent survey and treatment of recent developments is given by McLachlan and Krishnan (1997).

The EM algorithm interleaves two steps, starting from initial, assumed values for the missing data:

1. the E-step, in which the expected value of the likelihood is found with respect to the unknown values, using the current estimate of the parameters, conditioned on the observations.
2. the M-step, in which this expectation is maximized to yield a new set of parameters.

The E- and M-steps are iterated with each iteration guaranteed to increase the likelihood until we converge to a local maximum of the likelihood function. Convergence is proved by Dempster et al. (1977) and Wu (1983) among others. Like other optimisation techniques that find local maxima by gradient ascent, the particular local maximum found in general depends on the start point of the iteration—i.e., the assumed initial values of the missing data.

In the specific case of letter-phoneme alignment, the observed data are the words listed in the dictionary in terms of their paired spellings/pronunciations. The missing data are the parameters describing the probabilistic correspondence between words and letters that underlie the alignment process and that are compiled into matrix $\mathbf{A}$. As mentioned in Section 4 below, we maximize not the likelihood for word w at iteration k but the maximal DP score (as described in the previous section) given the association matrix from the iteration. Hence, the process must start with an association matrix $\mathbf{A}^0$ initialized with some appropriate values.

The simplest way to obtain $\mathbf{A}^0$ is the *naïve* initialization, found as follows. 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^0_{lp} of $\mathbf{A}^0$ is incremented. After the first pass through the dictionary, each element a^0_{lp} contains a count of the number of times letter l and phoneme p appear in the same word. This is not of course to say that a specific l and p do align; the rationale is that they can *only* align if they occur in the same word. Although we do not expect this to give a very good estimate of $\mathbf{A}$, an initial alignment can be attempted from $\mathbf{A}^0$.

Once we have this (imperfect) alignment, we can perform a second pass through the dictionary to produce a new and better association matrix $\mathbf{A}^1$ with elements a^1_{lp} that count the number of times letter l and phoneme p appear *at the same (aligned) position*, i. At this first iteration, nulls are now introduced into the dictionary as a consequence of the DP matching so that letters can associate with null phonemes and phonemes can associate with null letters. Although these nulls obviously affect the counts of letter-phoneme associations, they are not themselves entered as part of the updated matrix $\mathbf{A}^1$. They are omitted because to do so worked far better than including nulls. If we include nulls in the set of letters and phonemes at the EM stage, we are effectively building in an unnatural tendency for alignments to exploit nulls, because of their cumulative high scoring over a variety of situations. Hence, we restrict the role of the nulls to the DP matching stage.

Proceeding as above, a new set of candidate alignments can now be produced and scored, a new 'best' alignment again selected, and $\mathbf{A}^1$ updated to $\mathbf{A}^2$. Further iterations can then be used to improve the alignments, and the estimates of the association matrix, until convergence.

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 maximization step, this can be seen as an EM-like algorithm.

4. Issues with the Alignment Algorithm

Many interesting issues arise with respect to alignment based on the EM and DP algorithms. In this section, we briefly discuss the more important of them.

As a form of gradient ascent procedure, convergence is to a local maximum that in general depends upon the start point, i.e., the matrix $\mathbf{A}^0$. One possible start point uses the simple naïve approach of the previous section.

Intuitively, this has the disadvantage of allowing any letter to associate with any phoneme, no matter that one might appear at the beginning of a long word and the other at the end. Hence, an attractive possibility is to weight the entries a^0_{lp} inversely according to the difference of the position indices of the l and p symbols. For example, the position-index difference between letter h and phoneme /z/ of (*phase*, /feɪz/) is $|2 - 3| = 1$. Various weighting schemes could be envisaged. Yet another possibility is to use the manual alignments devised for training NETtalk (Sejnowski and Rosenberg, 1987) or NETspeak (McCulloch et al., 1987) to obtain $\mathbf{A}^0$. (In this latter case, the counts entered into $\mathbf{A}^0$ *will* have taken account of nulls.) Further, Black et al. (1998) have described a similar algorithm to ours in which they specify a set of "allowables", i.e., letters and phonemes that can plausibly associate on the basis of prior intuitive knowledge of letter-phoneme correspondences. This can be used to define binary values for a^0_{lp} (which become continuous on subsequent EM iterations). One of the major aims of this paper was to evaluate the wide variety of possibilities for initialization (see Section 5.2).

One very important issue is evaluating quantitatively the effectiveness of any alignment algorithm. However, this is difficult since there is no canonically correct 'gold standard' alignment in all cases (see Introduction). Scoring on the basis of human judgement is likely to be subjective and inconsistent between judges and is, in any case, not practical for the sort of very large dictionaries that we wish to use. Although it is possible (and indeed sensible) to have a human expert check obvious problem cases (e.g., *axe*, *know*, *phase*, . . .), and we did in fact do this during program development, it does not amount to a full and thorough evaluation, giving a global summary figure of merit. Thus, we have decided to assess our alignment results indirectly according to the number of words correctly transcribed by a pronunciation by analogy (PbA) system. For this purpose, we have used the PbA system of 2000.

Another issue is what we have previously called the 'harmonization' of the different phoneme inventories used by different researchers and/or dictionary compilers (Damper et al., 1999). Thus, if we wish to use the NETtalk manual alignment to estimate $\mathbf{A}^0$ in order to align a dictionary such as BEEP (see below), we must have some way of mapping the different sets of phonemes used by the different dictionaries onto a common set. Because our goal is to align BEEP, we obviously choose the BEEP symbols as the common set. Tables 1 and 2 show the harmonization scheme

Table 1. Harmonization scheme used to map the NETtalk phoneme set onto the BEEP set.

NETtalk	BEEP	as in . . .	IPA
a	aa	father	ɑ
b	b	bet	b
c	ao	bought	ɔ
d	d	dime	d
e	ey	bake	eɪ
f	f	fin	f
g	g	guess	ɡ
h	hh	head	h
i	iy	peat	i
k	k	kitten	k
l	l	let	l
m	m	met	m
n	n	net	n
o	ow	boat	oʊ
p	p	pet	p
r	r	red	r
s	s	set	s
t	t	test	t
u	uw	lute	u
v	v	vest	v
w	w	wet	w
x	ax	about	ə
y	y	yet	j
z	z	zoo	z
A	ay	bite	aɪ
C	ch	chin	tʃ
D	dh	this	ð
E	eh	bet	ε
G	ng	sing	ŋ
I	ih	bit	ɪ
J	jh	gin	ʤ
K	k s	sexual	k ʃ
L	l	bottle	ɫ
M	m	abysm	(ə)m
N	n	button	(ə)n
O	oy	boy	ɒɪ
Q	k w	quest	k w
R	er	bird	ɜ
S	sh	shin	ʃ
T	th	thin	θ
U	uh	book	ʊ
W	aw	bout	aʊ

(*Continue on next page.*)

Table 1. (*Continue*).

NETtalk	BEEP	as in ...	IPA
X	k s	sex	k s
Y	y uw	cute	j u
Z	zh	leisure	ʒ
@	ae	bat	a
!	t s	nazi	t s
#	g z	examine	ɡ z
+	w aa	bourgeois	w ɑ
*	w	whack	ʍ
ˆ	ah	but	ʌ

used to map the NETtalk and NETspeak phoneme sets onto BEEP. Note that BEEP uses a phoneme inventory of 44 symbols (excluding the null phoneme), whereas the NETtalk and NETspeak inventories are both of size 51 (again excluding the null phoneme).

The symbols listed in the 'NETtalk' column of Table 1 are those in the file downloaded from `http://www.speech.cs.cmu.edu/comp.speech` and not the ones tabulated in Appendix A of 1987. The downloaded file includes a symbol '+' which is not listed in the paper and excludes a symbol '|' which is listed in the paper. In general, harmonization can never be an exact process, because of idiosyncratic choice of phoneme inventories by the different individual compilers of the transcribed dictionaries, which often reflect dialectal differences. For instance, Sejnowski and Rosenberg (1987) use the same symbol /a/ to transcribe both the a vowel in *father* and the ɒ vowel in *stock*, as these are probably the same vowel for their dialect of American English. So the mapping from NETtalk to BEEP symbols is not one-to-one. We can only try to achieve the most consistent mapping according to our intuitions.

A final issue is that the EM algorithm is properly a probabilistic algorithm. We experimented with various normalizations, corresponding to various probabilistic models, but none performed as well as using simple (unnormalized) frequency counts directly from the association matrix **A**. Hence, all results presented here use this formulation. This is the reason we refer to our algorithm as "EM-like". The effect of using unnormalized counts (rather than proper probabilities) on convergence is unknown but, as we shall see, this did not prove to be an issue in practice.

Table 2. Harmonization scheme used to map the NETspeak phoneme set onto the BEEP set.

NETspeak	BEEP	as in ...	IPA
A	ax	about	ə
B	b	bet	b
D	d	dime	d
E	eh	bet	ɛ
F	f	fin	f
G	g	guess	ɡ
H	hh	head	h
I	ih	bit	ɪ
J	jh	gin	ʤ
K	k	kitten	k
L	l	let	l
M	m	met	m
N	n	net	n
O	oh	stock	ɒ
P	p	pet	p
R	r	red	r
S	s	set	s
T	t	test	t
U	ah	but	ʌ
V	v	vest	v
W	w	wet	w
Y	y	yet	j
Z	z	zoo	z
AA	ae	bat	a
AI	ey	bake	eɪ
AR	aa	father	ɑ
AW	ao	bought	ɔ
CH	ch	chin	tʃ
DH	dh	this	ð
EE	iy	peat	i
EI	ea	air	ɛə
ER	er	bird	ɜ
EY	ih	despite	ɪ
GZ	g z	examine	ɡ z
IA	ia	ear	ɪə
IE	ay	bite	aɪ
KH	k sh	anxious	k ʃ
KS	k s	sex	k s
KW	k w	quest	k w
NG	ng	sing	ŋ
OA	ow	boat	oʊ
OI	oy	boy	ɒɪ

(*Continue on next page.*)

Table 2. (*Continue*).

NETspeak	BEEP	as in ...	IPA
OO	uh	book	ʊ
OU	aw	bout	aʊ
SH	sh	shin	ʃ
TH	th	thin	θ
UL	l	bottle	ɫ
UR	ua	moor	ϖə
UU	uw	lute	u
YU	y uw	cute	j u
ZH	zh	leisure	ʒ

5. Results

In this section, we report the results of using our algorithm to align a large dictionary.

5.1. BEEP Dictionary

Our algorithm has been tested by using it to align BEEP: the British English Example Pronunciation dictionary. BEEP is publically accessible and can be downloaded from `http://www.speech.cs.cmu.edu/comp.speech`. It is typical of the size and content of the on-line dictionaries used for current speech technology applications. BEEP was constructed by amalgamating several public domain dictionaries to yield a large composite. The version used here contained 257,033 words. Note that there has been no strong quality control in constructing BEEP. Consequently, it contains several erroneous word entries (e.g., INDISPUTABLE for *indissoluble*, UNDILAPIDATED for *undiluted*) and transcriptions (e.g., for *abnegation*). Those that we discovered have been removed but we certainly cannot guarantee to have found all errors. We also removed all words with multiple pronunciations for conformity with the evaluation protocol in Marchand and Damper (2000). This gives a dictionary with 198,632 entries in all.

5.2. Initializations

The following initializations were used:

- naïve;
- a weighted scheme with $W = \beta/(1 + |d|)$ where d is the letter-phoneme position-index difference, and β is a heuristic scaling set to 40 for the results reported here;
- the NETtalk manual alignment (20,009 words);
- the NETspeak manual alignment (16,280 words);
- various random alignments.

5.3. Convergence

The convergence criterion was that there was no change as between $\mathbf{A}^k$ and $\mathbf{A}^{k-1}$.

Figure 2 shows the convergence behavior for the NETtalk initialisation. The quantity graphed is the total DP score for the whole dictionary at the end of iteration k, i.e., $S_k = \sum_{i=1}^{198,632} \text{align}^k(w_i)$. Note that convergence requires that the $\mathbf{A}$ matrix is unchanged between iterations, $\mathbf{A}^k = \mathbf{A}^{k-1}$, which (because nulls are not included in the $\mathbf{A}$ matrix) is not quite the same as the total DP score remaining unchanged, $S_k = S_{k-1}$. The total DP score at the zeroth iteration, S_0, is very low in this case, because only the 20,009 words of the originally-aligned NETtalk dictionary can be scored.

Figure 3 shows convergence behavior for two different initializations, excluding the total DP score at the zeroth iteration, S_0. This gives a clearer view of the convergence for the NETtalk initialization than does Fig. 2 where the very low value of S_0 swamps the trend. For the naïve initialization, it is not really sensible to depict S_0 anyway since the dramatic overcounting of associations (every letter is counted $|p_w|$ times and every phoneme is counted $|l_w|$ times) produces a very high score that is effectively meaningless. For both initializations, most of the improvement takes place between the first and second iterations. This was found to be a general characteristic of the results. For all initializations, convergence was achieved in between 5 to 8 iterations. The manual alignment of the NETtalk dictionary, even though it is much smaller than BEEP, shows a clear benefit in terms of a higher score at iteration 1 together with faster convergence. The score at convergence, S_C, was remarkably consistent across the various initializations, suggesting that the search problem is strongly convex. The best value obtained was $S_C = 8.579 \times 10^{10}$ for the NETtalk initialization whereas the worst value was $S_C = 8.473 \times 10^{10}$ for one of the random initializations. Generally, the random initialization values were slightly lower than the others.

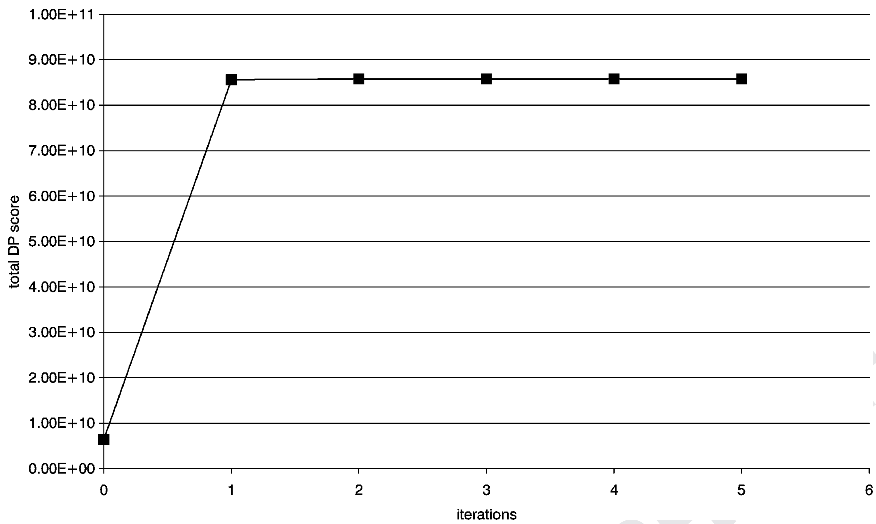

Figure 2. Convergence behavior of the alignment algorithm for the NETtalk initialization.

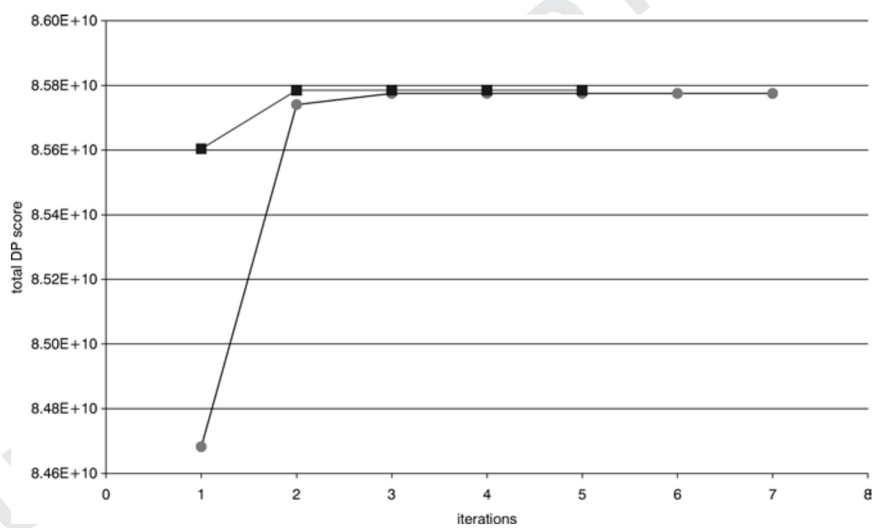

Figure 3. Convergence behavior of the alignment algorithm for two different initializations. Rectangles: NETtalk initialization; Circles: naïve initialization.

5.4. Analysis of Association Matrices

Figures 4(a) and (b) show the association matrices for the naïve initialization initially, $\mathbf{A}^0$, and at convergence, $\mathbf{A}^7$. The larger association values in Fig. 4 are a consequence of the overcounting mentioned above. As expected, the matrix is considerably less random (i.e., peakier) at convergence. Quantitatively, the (negative) entropy of the $\mathbf{A}^0$ matrix was 8.84 bits whereas that of the converged matrix was 5.24 bits; these figures compare with 10.13 bits for the equiprobable case. Encouragingly, the strongest peaks at convergence, corresponding to the major letter-phoneme associations, are also among the strongest peaks in $\mathbf{A}^0$, indicating that the naïve initialization, albeit very simple, still provides an effective start point for our algorithm.

There is a wealth of information about letter-phoneme correspondences in English to be gleaned from the **A** matrix obtained at convergence. Since nulls are introduced into the aligned dictionary only at the DP matching stage (see Section 3) and do not figure in the **A** matrix, they are not considered explicitly in

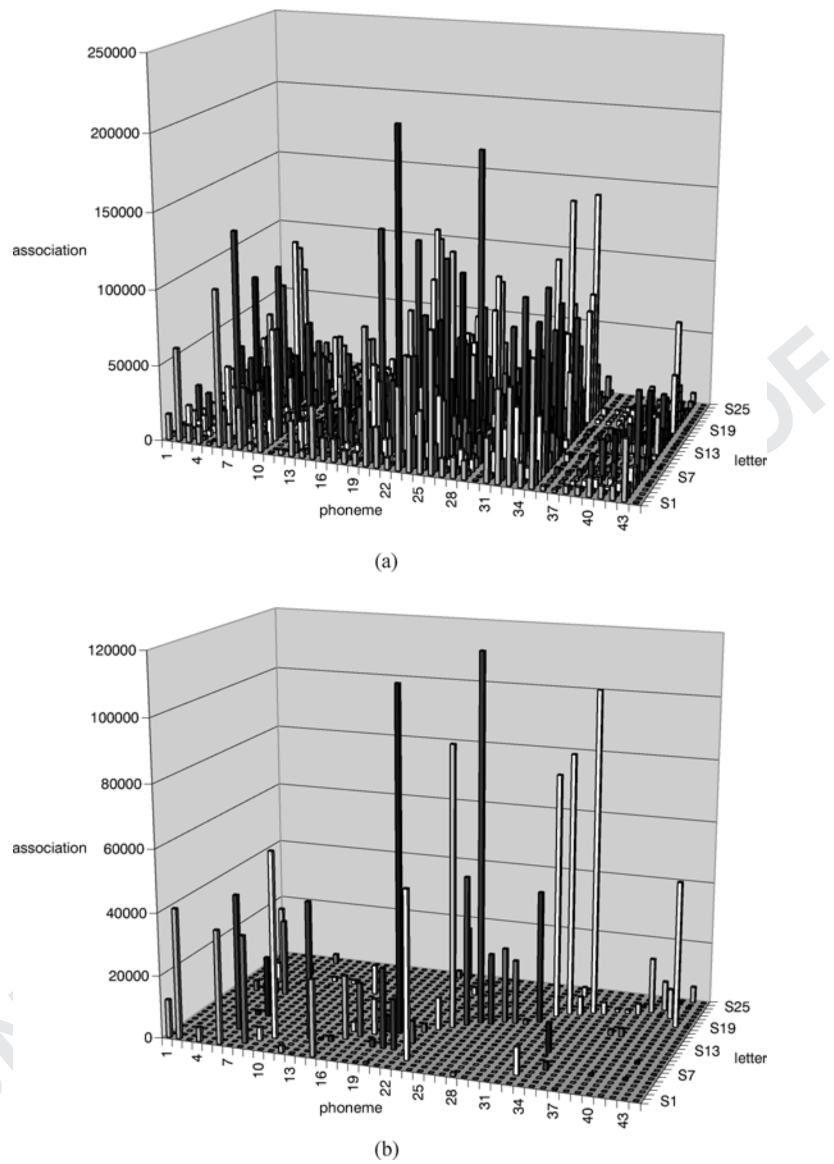

Figure 4. Association matrices for the naïve initialization both initially, $\mathbf{A}^0$, and at convergence, $\mathbf{A}^7$.

the remarks that follow. With this proviso, the commonest correspondence overall was $n \rightarrow$ /n/. The commonest letter participating in correspondences is *i*, which occurs 148,913 times in the matrix. This is slightly surprising as the commonest letter overall is *e*. The apparent discrepancy is explained by the number of times letter *e* participates in a functional spelling unit such as *ea* and so aligns with null (with the letter *a* aligning with the vowel phoneme). The least common letter participating in correspondences is *q*, which occurs just 17 times. Again, *q* almost invariably occurs in a *qu* functional spelling unit, with *q* aligning with a null phoneme, which reduces its count in the matrix. The commonest phoneme is /ɪ/ at 138,176 occurrences, which can be understood from the frequency with which letter *i* occurs and the fact that $i \rightarrow$ /ɪ/ is a very common correspondence (at 109,508 occurrences). Schwa, /ə/, is relatively less common than /ɪ/ at 190,975 occurrences. Intuitively, one might expect schwa to be the commonest vowel, but it is perhaps more likely than /ɪ/ to align with a null letter. Of the letters, *o* displays most variability in its association with phonemes, with no less than nine correspondences with a frequency count of 1000 or more. The least variability is shown by letter *m*, which almost always associates with phoneme /m/. Schwa displays easily the most variability in its association with letters, participating in five correspondences (with letters *a*, *e*, *i*, *o* and *u*) with a count greater than 1000. The least variable phoneme was /ŋ/, which associated with letter *n* in all but just 2 cases.

5.5. *Assessing Alignment Performance Using PbA*

As previously stated, alignment results were assessed using PbA. Each word was removed from the dictionary and a pronunciation determined from the word's spelling by analogy with all other words. The Marchand and Damper PbA system uses multiple (actually five) criteria to select between candidate pronunciations to find the 'best'. There is, however, a problem in that PbA was designed to transcribe text in which there will obviously be no null letters. Yet here, null letters have been added to the alignments of many words. Our first step, then, has been to ignore any words with null letters, reducing the number of words to be tested from 198,632 to approximately 177,000. (The number varies with the exact initialization used.) This is an obvious simplification of the problem, but should nonetheless yield interesting insights.

Table 3 shows results obtained (for words without null letters) in terms of words and phonemes correctly pronounced for each of the initializations used. Several different random initializations were used, but results were very similar and so figures for one only are tabulated here. In each case, we show the results for the best single scoring criterion of the five, for the best combination, and when all five are combined. Note that 10100 in the column heading indicates that scoring strategies 1 and 3 as described by Marchand and Damper (2000, pp. 207–208) provided the best combination performance for all initializations. Although space precludes a full description of our PbA methodology, we mention that strategy 1 takes the product of arc frequencies along the shortest path in the pronunciation lattice, whereas strategy 3 counts the number of identical pronunciations having the same shortest path length. Strategy 1 is relatively popular in PbA (e.g., Damper and Eastmond, 1997) whereas we are not aware that any other researchers have ever used strategy 3, which interestingly turns out to be best performing single strategy overall.

Table 3. Results when alignment of the BEEP dictionary is assessed by the performance of a pronunciation by analogy system, for various initializations. Words with null letters in their alignments have been ignored at this stage.

	Best Single	Best Combination	All 5
NAÏVE	00100	10100	11111
Words (%)	85.84	87.32	85.96
Phonemes (%)	97.52	97.78	97.57
W WEIGHTED	00100	10100	11111
Words (%)	85.87	87.36	86.00
Phonemes (%)	97.60	97.85	97.65
NETTALK	00100	10100	11111
Words (%)	86.00	87.41	86.05
Phonemes (%)	97.59	97.83	97.63
NETSPEAK	00100	10100	11111
Words (%)	86.01	87.48	86.11
Phonemes (%)	97.64	97.89	97.70
RANDOM	00100	10100	11111
Words (%)	85.87	87.38	85.69
Phonemes (%)	97.51	97.78	97.57

The figures in Table 3 are remarkably consistent, indicating that the particular initialization used does not have a dramatic effect. This is in spite of our attempts to restart the algorithm from a variety of very different points, suggesting that the search space is strongly convex. It is worth noting, however, that as a consequence of the large dictionary size (approximately 177,000 words) the difference between the best Best Combination of 87.48% (for the NETtalk initialization) and the worst Best Combination of 87.32% (for the naïve initialization) is in fact marginally significant at the 5% level (binomial test, $z = 2.026$, $p \sim 0.021$).

The best PbA performance is found for NETspeak but initializing alignment with the NETspeak dictionary actually produced a slightly lower total DP score at convergence than initializing with NETtalk. In other words, the total DP score at convergence is a good but not perfect indicator of PbA performance. Examination of the final alignments revealed that these were strongly similar; there were typically somewhere between 10 and 100 different alignments only between one initialization and another. Most often, differences were due to the specific placement of nulls in words having many silent letters (e.g., *bourgeoisie*, *heavyweight*, *memoirs*). Frequently, these were words of foreign (French) origin.

This is certainly among the best performance figures ever reported on English letter-phoneme conversion, in terms of word-level accuracy on a large dictionary. Previously (Damper et al., 1999), we obtained 71.8% words correct using PbA on a much smaller dictionary—the 16,280 manually-aligned words used by McCulloch et al. (1987) to train NETspeak. (It should be noted, however, that BEEP uses a smaller phoneme inventory of 44 symbols than the 51 used in the NETspeak dictionary, making for a somewhat easier problem.) A further observation is that using all five strategies does not give best performance, as it did for our earlier work with smaller dictionaries (Marchand and Damper, 2000). In assessing performance, however, we must remember that we have simplified the problem by ignoring words with nulls, which arguably gives a too optimistic view of the present results. However, even under the maximally pessimistic assumption that PbA were to get *all* the words with null letters wrong, the 85.8% words correct for best single strategy, naïve start point, would fall to 76.1%—still a very respectable result on such a sizable dictionary.

To gain further insight into this issue, PbA was used to produce pronunciations for all 198,632 words including those with null letters in their alignment, treating the latter as a legitimate input symbol (even though it never could be in practice). Results for the best combination averaged 82.3% words correct, showing that high accuracy is potentially achievable if only 'missing' nulls in the PbA input could be appropriately introduced.

6. Discussion and Conclusions

We have described a form of the EM algorithm, used with dynamic programming to align a dictionary of word spellings and their pronunciations. Such alignment problems commonly occur in speech technology and natural language processing. The issues that arise in solving this important problem have been detailed and discussed. The quality of the obtained alignment has been assessed using pronunciation by analogy to derive pronunciations for all words in the dictionary from their spelling, using the aligned data as a knowledge base. Since the EM algorithm is effectively a gradient ascent procedure prone to finding local maxima, alignment has been performed from a variety of initializations, or start points. Results are judged to be extremely encouraging, and are relatively insensitive to a wide variety of start points. This indicates that the search space is strongly convex and, hence, that local maxima are not a practical problem.

Our work has several similarities with that of Ristad and Yianilos (1998). This is perhaps not surprising as they take the topic of stochastic transduction as their motivation, whereas the ideas reported in this paper had their early expression in our own work, which led to the use of stochastic transduction to solve problems in TTS conversion, including letter-phone alignment (Luk and Damper, 1996, 1998). Ristad and Yianilos also use dynamic programming in conjunction with the EM algorithm to learn edit distances between two strings. Since the string edit operations of insertion and deletion can be interpreted as the introduction of nulls into one string or another—either the word's spelling or its pronunciation—there is clearly a strong relation between the two pieces of work. As Jansche (2001) writes: "The problem of letter-to-sound conversion is very similar to the problem of modeling pronunciation variation". However, although Ristad and Yianilos consider the problem of pronunciation modelling in speech technology, they do not consider alignment problems as such.

This work represents the most comprehensive study to date of letter-phoneme alignment, at the same time achieving what is probably the best reported performance on the difficult task of letter-phoneme conversion of unknown words of English. Since the aligned BEEP dictionary is a potentially valuable resource, the version obtained from the NETspeak initialization (which produced best performance on letter-phoneme conversion) is made freely available for research use at `http://festvox.org/packed/data/damper`. Since our software has wide applicability, we are also working to provide an on-line facility at which researchers can submit dictionaries for alignment.

References

Abercrombie, D. (1981). Extending the Roman alphabet: Some orthographic experiments of the past four centuries. In R.E. Asher and E. Henderson (Eds.), *Towards a History of Phonetics*. Edinburgh, UK: Edinburgh University Press, pp. 207–224.

Bagshaw, P.C. (1998). Phonemic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexicon compression. *Computer Speech and Language*, *12*(2):119–142.

Baum, L.E. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In *Inequalities III: Proceedings of the Third Symposium on Inequalities*, Los Angeles, CA, pp. 1–8.

Bellman, R. (1957). *Dynamic Programming*. Princeton, NJ: Princeton University Press.

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

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

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

Coltheart, M. (1978). Lexical access in simple reading tasks. In G. Underwood (Ed.), *Strategies of Information Processing*. New York: Academic Press, pp. 151–216.

Coltheart, M. (1984). Writing systems and reading disorders. In L. Henderson (Ed.), *Orthographies and Reading: Perspectives from Cognitive Psychology, Neuropsychology and Linguistics*. London, UK: Lawrence Erlbaum Associates, pp. 67–79.

Damper, R.I. (Ed.) (2001). *Data-Driven Methods in Speech Synthesis*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Damper, R.I. and Eastmond, J.F.G. (1996). Pronouncing text by analogy. In *Proceedings of 16th International Conference on Computational Linguistics*. Copenhagen, Denmark, Vol 2, pp. 268–273.

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

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

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

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

Federici, S., Pirrelli, V., and Yvon, F. (1995). Advances in analogy-based learning: False friends and exceptional items in pronunciation by paradigm-driven analogy. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'95) Workshop on New Approaches to Learning for Natural Language Processing*, Montreal, Canada, pp. 158–163.

Forney, G. D. (1973). The Viterbi algorithm. In *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278.

Hartley, H. (1958). Maximum likelihood estimation from incomplete data. *Biometrics 14*, 174–194.

Henderson, L. (1984). Writing systems and reading processes. In L. Henderson (Ed.), *Orthographies and Reading: Perspectives from Cognitive Psychology, Neuropsychology and Linguistics*. London, UK: Lawrence Erlbaum Associates, pp. 11–24.

Hopcroft, J.E., Motwani, R., and Ullman, J.D. (2001). *Introduction to Automata Theory, Languages, and Computation*, 2nd ed. Boston, MA: Addison-Wesley.

Jansche, M. (2001). Re-engineering letter to sound rules. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics, NAACL 2001*, Pittsburg, PA. Paper N01-1015 in on-line archive at `http://acl.ldc.upenn.edu/N/N01/`.

Katz, L. and Feldman, L.B. (1981). Linguistic coding in word recognition: comparisons between a deep and a shallow orthography. In A.M. Lesgold and C.A. Perfetti (Eds.), *Interactive Processes in Reading*. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 85–106.

Knill, K. and Young, S. (1997). Hidden Markov models in speech and language processing. See Young and Bloothooft (1997), pp. 27–68.

Kruskal, J.B. (1983). An overview of sequence comparison. In D. Sankoff and J. B. Kruskal (Eds.), *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*. Reading, MA: Addison-Wesley, pp. 11–44.

Lawrence, S.G.C. and Kaye, G. (1986). Alignment of phonemes with their corresponding orthography. *Computer Speech and Language*, *1*(2):153–165.

Liberman, I., Liberman, A., Mattingly, I. and Shankweiler, D. (1980). Orthography and the beginning reader. In J. Kavanagh and R. Venezky (Eds.), *Orthography, Reading and Dyslexia*. Baltimore, OH: University Park Press, pp. 137–153.

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

Luk, R.W.P. and Damper, R.I. (1992). Inference of letter-phoneme correspondences by delimiting and dynamic time warping techniques. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'92*. San Francisco, CA, Vol 2, pp. II.61–II.64.

Luk, R.W.P. and Damper, R.I. (1993). Inference of letter-phoneme correspondences with pre-defined consonant and vowel patterns. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'93*. Minneapolis, MN, Vol 2, pp. II.203–II.206.

Luk, R.W.P. and Damper, R.I. (1996). Stochastic phonographic transduction for English. *Computer Speech and Language*, *10*(2):133–153.

Luk, R.W.P. and Damper, R.I. (1998). Computational complexity of a fast Viterbi decoding algorithm for stochastic letter-phoneme transduction. *IEEE Transactions on Speech and Audio Processing*, *6*(3):217–225.

Marchand, Y. and Damper, R.I. (2000). A multistrategy approach to improving pronunciation by analogy. *Computational Linguistics*, *26*(2):195–219.

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

McLachlan, G.J. and Krishnan, T. (1997). *The EM Algorithm and Extensions*. New York, NY: John Wiley.

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

Needleman, S.B. and Wunsch, C.D. (1970). An efficient method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, *48*(3):444–453.

Neuhoff, D.L. (1975). The Viterbi algorithm as an aid in text recognition. In *IEEE Transactions on Information Theory*, IT-21:222–226.

Parfitt, S.H. and Sharman, R.A. (1991). A bidirectional model of English pronunciation. In *Proceedings of 2nd European Conference on Speech Communication and Technology, Eurospeech'91*. Genova, Italy, Vol 2, pp. 800–804.

Pirrelli, V. and Federici, S. (1994). On the pronunciation of unknown words by analogy in text-to-speech systems. In *Proceedings of the*

Second Onomastica Research Colloquium. London, UK, pp. 43–50.

Pirrelli, V. and Federici, S. (1995). 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*. Madrid, Spain, Vol 1, pp. 855–858.

Pirrelli, V. and Yvon, F. (1999). The hidden dimension: A paradigmatic view of data-driven NLP. *Journal of Experimental and Theoretical Artificial Intelligence*, *11*(3):391–408.

Ristad, E.S. and Yianilos, P.M. (1998). Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(5):522–532.

Sampson, G. (1985). *Writing Systems*. London, UK: Hutchinson.

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

Sullivan, K.P.H. (2001). Analogy, the corpus and pronunciation. See 2001, pp. 45–70.

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

Turvey, M.T., Feldman, L.B. and Lukatela, G. (1984). The Serbo-Croatian orthography constrains the reader to a phonologically analytic strategy. In L. Henderson (Ed.) *Orthographies and Reading: Perspectives from Cognitive Psychology, Neuropsychology and Linguistics*. London, UK: Lawrence Erlbaum Associates, pp. 81–89.

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

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

Viterbi, A.J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, *IT-13*(2):260–269.

Wu, C.F.J. (1983). On the convergence properties of the EM algorithm. *Annals of Statistics*, *11*(1):95–103.

Young, S. and G. Bloothooft (Eds.) (1997). *Corpus-Based Methods in Language and Speech Processing*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

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

Yvon, F. (1996b). *Prononcer par Analogie: Motivations, Formalisations et Évaluations*. PhD thesis, ENST, Paris, France.