Can syllabification improve pronunciation by analogy of English? Nat Lang Eng

Article *in* Natural Language Engineering · March 2007

DOI: 10.1017/S1351324905004043 · Source: DBLP

Robert I. Damper
University of Southampton
393 PUBLICATIONS **3,584** CITATIONS

Natural Language Engineering **13** (1): 1–24. © 2006 Cambridge University Press
doi:10.1017/S1351324905004043 First published online 10 January 2006 Printed in the United Kingdom

Can syllabification improve pronunciation by analogy of English?

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

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

(*Received 23 July 2004; revised 9 September 2005*)

Abstract

In spite of difficulty in defining the syllable unequivocally, and controversy over its role in theories of spoken and written language processing, the syllable is a potentially useful unit in several practical tasks which arise in computational linguistics and speech technology. For instance, syllable structure might embody valuable information for building word models in automatic speech recognition, and concatenative speech synthesis might use syllables or demisyllables as basic units. In this paper, we first present an algorithm for determining syllable boundaries in the orthographic form of unknown words that works by analogical reasoning from a database or corpus of known syllabifications. We call this *syllabification by analogy* (SbA). It is similarly motivated to our existing *pronunciation by analogy* (PbA) which predicts pronunciations for unknown words (specified by their spellings) by inference from a dictionary of known word spellings and corresponding pronunciations. We show that including perfect (according to the corpus) syllable boundary information in the orthographic input can dramatically improve the performance of pronunciation by analogy of English words, but such information would not be available to a practical system. So we next investigate combining automatically-inferred syllabification and pronunciation in two different ways: the series model in which syllabification is followed sequentially by pronunciation generation; and the parallel model in which syllabification and pronunciation are simultaneously inferred. Unfortunately, neither improves performance over PbA without syllabification. Possible reasons for this failure are explored via an analysis of syllabification and pronunciation errors.

1 Introduction

The syllable has been much discussed as a linguistic unit. Whereas some linguists make it central to their theories (e.g., Pulgram 1970; Selkirk 1982), others have ignored it or even argued against it as a useful theoretical construct (e.g., Kohler 1966). Much of the controversy centres around the difficulty of defining the syllable. Crystal (1980, p. 342) states that the syllable is '[a] unit of pronunciation typically larger than a single sound and smaller than a word' but goes on to write: 'Providing a precise definition of the syllable is not an easy matter, and there are several theories in both phonetics and phonology that have tried to clarify matters'. For instance, Blevins (1995, p. 207) writes: 'The syllable is the phonological unit which organizes segmental melodies in terms of sonority'. However, none of these theories is universally accepted. There is general agreement that a syllable consists of a *nucleus* which is almost always a vowel, together with zero or more preceding consonants (the *onset*) and zero or more following consonants (the *coda*) but determining exactly which consonants of a multisyllabic word belong to which syllable is problematic. There can be interactions between morphological and syllabic structure. For instance, Daelemans and van den Bosch (1992, p. 28) write that 'Dutch syllabification is an interesting problem . . . because the process involves phonological and morphological constraints that are sometimes conflicting' and according to Kiraz and Möbius (1998, p. 73): 'In both English and German, morpheme boundaries frequently override default syllabification in compounded words'. Good general accounts of the controversy from a theoretical perspective are provided by Treiman and Zukowski (1990) and Goslin and Frauenfelder (2000), with the former more specifically considering English – the language of interest in this paper – and the latter focusing on French.

However it is defined, and regardless of the rights or wrongs of theorising about its linguistic status, native speakers of a language are easily able to count the syllables of a word based entirely on intuition. If the syllable does act as a structuring device in composing words, as many believe, then knowledge of this structure could well aid word modelling in automatic speech recognition (Greenberg 1999), the unit selection and composition process of concatenative synthesis, and other important tasks in natural language processing and speech technology. For example, van Santen, Shih, Möbius, Tzoukermann and Tanenblatt (1997) have shown that location within a syllable affects the duration of a phone, and Müller (2002, p. 37) writes: 'The correct pronunciation of a . . . word is not only dependent on the correct identification of phonemes but also on the correct assignment of syllables'. That is, the pronunciation of a phoneme can depend upon where it is in a syllable. So, arguments about the theoretical status of the syllable as a linguistic unit notwithstanding, there are good practical, engineering reasons for seeking powerful algorithms to syllabify words.

Traditional approaches to automatic syllabification have been knowledge-based, implementing notions such as the maximal onset principle (Pulgram 1970; Kahn 1976) and sonority hierarchy (Clements 1988; Blevins 1995), including ideas about what constitute phonotactically legal sequences in the coda, for instance. (But see Kessler and Treiman 1997, p. 295 who argue for replacing the notion of 'absolute, inviolable restrictions' on allowable phoneme sequences by a probabilistic account.) Other putative principles of syllabification include the idea that consonants more readily associate with, or are 'attracted to', stressed vowels (Hoard 1971). Frequently, the constraints implied by the different principles cannot be simultaneously satisfied and some means has to be found to arbitrate between the competing hypotheses. For instance, Goslin and Frauenfelder (2000) document the extensive disagreement between five syllabification procedures on approximately 23,000 French words: They write 'relatively small and simple differences in syllabification rules can reap large differences when applied across a wide range of stimuli' (p. 419). Unfortunately, it is far from clear how the arbitration should be done, i.e., we do not actually possess the requisite knowledge to determine a canonically correct syllabification. Although it could be argued (as do Goslin and Frauenfelder, p. 420) that human syllabification must be the *gold standard*, the extensive disagreement observed between human subjects in syllabification experiments makes this impractical. It is one thing for people to be able to count syllables, which amounts to recognising nuclei, but quite another to decide where the boundaries are. Further, it is unrealistic to expect to obtain human syllabifications for the tens or hundreds of thousands of words found in the dictionary of a typical text-to-speech (TTS) system or automatic speech recogniser.

In this work, we present an alternative to the knowledge-based approach which uses analogical reasoning from an evidence base, namely a dictionary of already-segmented words. We call this *syllabification by analogy* (SbA). We then use SbA to explore the potential that syllabification offers to improve automatic pronunciation in applications such as TTS synthesis. SbA is an example of an increasing popular approach to computational linguistics and speech technology, based on lazy learning (Aha, Kibler and Albert 1991; Aha 1997). By *lazy learning*, we mean a data-driven method that deliberately avoids the wholesale replacement of the example dataset by some compressed representation of its major regularities – so called *eager learning*. While lazy learning is exemplified by nearest-neighbour classification, a typical form of eager learning is the well-known back-propagation algorithm for training artificial neural networks, in which the training data are encoded into a small set of connection weights and thresholds.

Lazy learning has many advantages for natural language processing tasks including imposition of conformance with real data, avoidance of the costly building and maintenance of a complex knowledge base (often tainted by biases as to what we *actually* know), avoidance of the high computational cost of eager learning, excellent transparency of the algorithm and its results, good transferability of the method to different languages and to different tasks and, last but not least, excellent accuracy gained in large part by retaining 'exceptional' items in the evidence base (Daelemans, van den Bosch and Zavrel 1999). In fact, exceptions are much more important in language than is generally thought, where there has been a regrettable tendency to over-emphasise regularities and rules (e.g., Pinker 1999). The rationale seems to be that exceptions are rare, yet individually rare events are so characteristic of language that they collectively occur in large numbers. This has been called the LNRE (large numbers of rare events) phenomenon by Baayen (2001). Correct treatment of the LNRE phenomenon is crucial to obtaining high accuracy in language processing tasks.

There is a small existing literature on data-driven syllabification. Although the training data will be language-dependent, the approach is applicable to any language for which appropriate machine-readable data exist. Daelemans and van den Bosch (1992) compare various methods for Dutch and show that the generalisation performance of back-propagation learning is not better than symbolic (knowledge-based) approaches and that both are inferior to a form of lazy learning that they call exemplar-based generalisation, previously described by Weijters (1991). They report a best result of 98.3% patterns correct, where a *pattern* is a fixed-size (7-character) string and the classification task is to determine if the central character (i.e., the 4th one of the 7) is the first letter of a syllable. Although not mentioned in the paper, the size of the test set was only 1,945 words—the same number as used to test generalisation performance of the back-propagation learning (Daelemans, personal communication). Also for Dutch, Bouma (2003) describes a 'hyphenation' method but since the 'core rule of Dutch hyphenation is that hyphenation points fall between syllables' (p. 5), this is effectively a syllabification algorithm also. (Note too that, apart from one passage in their paper, Daelemans and van den Bosch use 'hyphenation' and 'syllabification' interchangeably.) Using a simple over-generative rule-based method which is then subjected to transformation-based learning (Brill 1995) to control the errors, Bouma reports an accuracy of 99.35% hyphens correct on a test set of approximately 29,000 words after training on approximately 260,000 words. Kiraz and Möbius (1998) present a probabilistic syllabification algorithm for English and German in which observed frequencies of onsets, nuclei and codas are converted into the weights of a weighted finite-state transducer. Unlike the approaches of Daelemans and van den Bosch and of Bouma which operate in the orthographic domain, Kiraz and Möbius's method works in the phoneme ('phonological') domain. Unfortunately, they do not give any quantitative evaluation data. Müller, Möbius and Prescher (2000) describe an approach in which a form of the expectation-maximisation (EM) algorithm is used to cluster example data in 3- and 5-dimensional syllable classes. (The 3-dimensional data are onset, nucleus and coda; the 5-dimensional data add position of syllables in the word and stress type.) Interestingly for our purposes, they then use this as a component part of a letter-to-phoneme conversion system for German. The overall system uses a hand-crafted rule-based system (see Müller 2001) which produces (all possible) candidate pronunciations. In this case, the method syllabifies a phoneme sequence. These sequences are then ranked by the probabilistic syllable model and the most probable analysis selected to give the predicted (syllabified) pronunciation. There are, however, several problems with this work from our perspective (apart from the use of hand-crafted rules). First, Müller *et al.* remove from their data set foreign words, acronyms, proper names, verbs and words of more than three syllables! Second, their test set was just 1,835 words, whereas we have argued (Damper, Marchand, Adamson and Gustafson 1999) for using the largest possible test set. Finally, they do not compare their pronunciation system that uses syllabification information with one which does not, so they are unable to assess the impact of syllable information on performance.

Here, we aim to assess the part that syllabification can play in pronunciation generation by analogy for English. We will consider both manual and automatic syllabification. It should be obvious from the above discussion that the latter is not an easy task. As well as the difficulty of precise definition of the syllable, there is ambiguity over whether the process operates in the orthographic or phonological domain (or both), over the role of morphology, and – if working in the orthographic domain – the exact relation to hyphenation in any given language. Further, in the absence of a sound linguistic theory of the syllable, it is not clear what should count as a correct output from a syllabification algorithm. If (as here) we are reasoning analogically from a set of pre-syllabified examples, how can we know that our evidence base is correct? The approach we adopt is that which has become established in machine learning in general. In the absence of a gold standard, and to make progress initially, we simply assume the correctness of our evidence base at the outset, and use it both to infer syllabifications for unknown words and for evaluation. It must then be understood that our results are only as good as the evidence base. Subsequently, we can seek confirmation or denial of the quality of these data by some means. We will return to discuss this point when concluding.

The remainder of this paper is structured as follows. Section 2 describes our existing *pronunciation by analogy* (PbA) algorithm which predicts pronunciations for unknown words, specified by their spelling, by inference from a dictionary of known word spellings and corresponding pronunciations – since our *syllabification by analogy* (SbA) program is a variation on the same idea. Thereafter, in section 3, we describe the evidence base we have used to infer both pronunciations and syllabifications; it is a manually-syllabified version of Webster's dictionary. Next, in section 4, we detail the SbA algorithm. We then show that perfect (according to the corpus) syllabification of the orthographic input to a pronunciation system can improve performance by a very considerable margin. The challenge then is to infer unknown syllable boundaries in such a way as to profit from this potential. In section 5, we present two attempts at doing so, neither of which actually improves on PbA (without syllabification). It seems that whereas good orthographic syllabification can help pronunciation, less than perfect syllabification can be harmful. Possible reasons for this are explored via an analysis of syllabification and pronunciation errors. Finally, we conclude with discussion of our results and pointers to future work in section 6.

2 Pronunciation by analogy

Pronunciation by analogy is a data-driven technique for converting spelling to sound that is attracting increasing attention as an automatic pronunciation method for text-to-speech (TTS) synthesis (e.g., Dedina and Nusbaum 1991; Sullivan and Damper 1993; Federici, Pirrelli and Yvon 1995; Damper and Eastmond 1997; Yvon 1996; Marchand and Damper 2000; Sullivan 2001). This has been driven by accumulating evidence that PbA easily outperforms traditional linguistic rewrite rules as used extensively in earlier TTS systems plus a variety of other data-driven methods for spelling-to-sound conversion (Damper *et al.* 1999).

PbA exploits the phonological knowledge inherent in a dictionary of words and their corresponding pronunciations. The underlying idea is that a pronunciation for an unknown word is derived by matching substrings of the input to substrings of known words in a lexicon (in this paper, we will use the terms *lexicon*, *dictionary*, *corpus*, *database* and *evidence base* interchangeably), hypothesising a partial pronunciation for each matched substring from the phonological knowledge, and assembling the partial pronunciations to form a final output. A seminal and

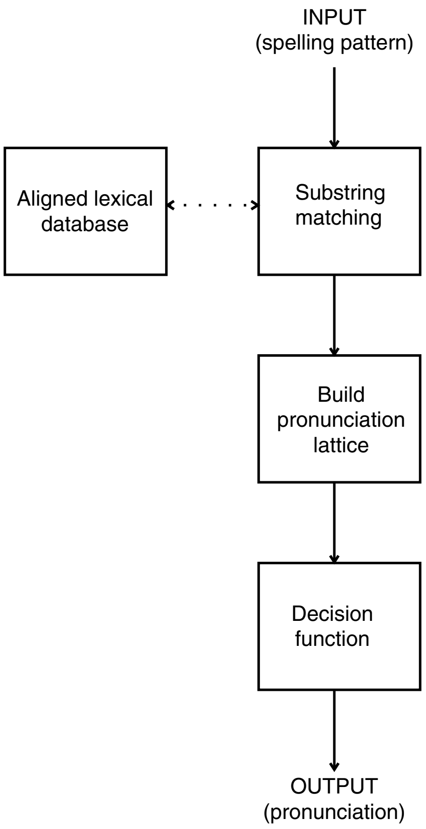

Fig. 1. Schematic of Dedina and Nusbaum's PRONOUNCE.

still typical PbA program is PRONOUNCE, described by Dedina and Nusbaum (1991), hereafter D&N. We will now present it in some detail, not only because it forms the basis for our own PbA algorithm, but also because our syllabification methods are based on it.

PRONOUNCE consists of four components (Fig. 1): the lexical database, the pattern matcher which compares the target input to all the words in the database, the pronunciation lattice (a data structure representing possible pronunciations), and the decision function, which selects the best pronunciation among the set of possible ones.

2.1 Pattern (substring) matching

In PRONOUNCE, an incoming word is matched in turn against all orthographic entries in the lexicon. For a given dictionary entry, the process starts in the D&N formulation with the input string $\mathscr{I}$ and the dictionary entry $\mathscr{D}$ left-aligned. Substrings sharing contiguous, common letters in matching positions in the two strings are then found. Information about these matching letter substrings – and their corresponding phoneme substrings in the dictionary entry under consideration – is entered into the pronunciation lattice as detailed immediately below. Note that this requires the

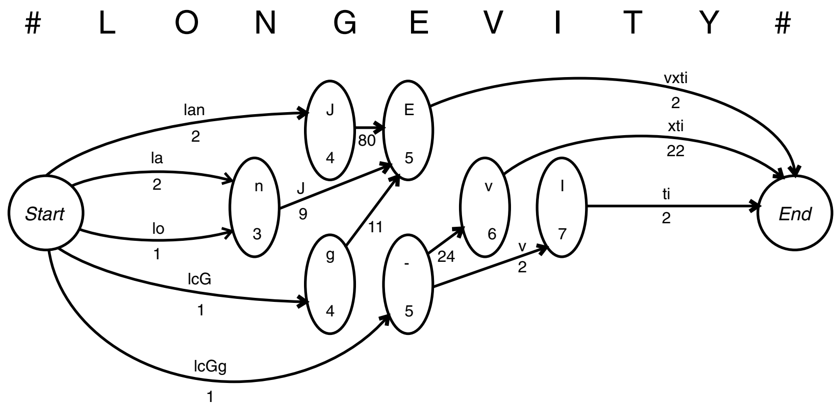

Fig. 2. Example of (simplified) pronunciation lattice for <longevity>. For clarity, only a subset of the arcs is shown, namely those corresponding to the shortest (length-3) paths. The phonetic symbols shown are those of Sejnowski and Rosenberg – see section 3.

letters and phonemes of each word in the lexicon to have been previously aligned in one-to-one fashion (Fig. 1) so that one or more partial pronunciations can be attributed to each matching substring. The shorter of the two strings is then shifted right by one letter and the matching process repeated. This continues until (in the D&N formulation) the two strings, $\mathscr{I}$ and $\mathscr{D}$, are right-aligned.

2.2 Pronunciation lattice

Matched substrings, together with their corresponding phonemic mappings as found in the (aligned) lexicon, are used to build the pronunciation lattice for the input string. A node of the lattice represents a matched letter, L_i, at some position, i, in the input. The node is labelled with its position index i and with the phoneme which corresponds to L_i in the matched substring, P_{im} say, for the mth matched substring. Two nodes, *Start* and *End* have special status; they represent the implicit spaces preceding and following the word, and match with the special word boundary marker '#' added before and after the letter string. An arc is placed from node i to node j if there is a matched substring starting with L_i and ending with L_j. The arc is labelled with the phonemes intermediate between P_{im} and P_{jm} in the phoneme part of the matched substring. Additionally, arcs are labelled with a frequency count (this is D&N's term, and is nothing to do with frequency of usage in written or spoken communication) that is incremented by one each time that substring with that pronunciation is matched during the pass through the lexicon.

Figure 2 shows an example pronunciation lattice for the word <longevity>. For clarity, only a subset of the arcs is shown – those contributing to the shortest path, since this is important to the decision on pronunciation, as described immediately below. The lattice was built by removing <longevity> from Webster's dictionary and then matching it against the remaining entries. This *leave-one-out* strategy is a very convenient way to derive pronunciations for all the words in a given dictionary, and thereby to assess performance; it is also known as k-fold cross validation, where k is the size of the dictionary. The candidate pronunciations are not necessarily distinct. Different shortest paths can obviously correspond to the same phoneme string, as in this example.

2.3 Decision function

A possible pronunciation for the input string then corresponds to a complete path through its lattice, from *Start* to *End*, with the output string assembled by concatenating the phoneme labels on the nodes/arcs in the order that they are traversed. The different paths are then scored so as to allow a decision as to which is the 'best'. This uses two heuristics in the original PRONOUNCE:

1. If there is a unique shortest path, then the pronunciation corresponding to this path is taken as the output.
2. If there is more than one shortest path, then the pronunciation corresponding to the best scoring of these is taken as the output.

In D&N's original work, the score used in Heuristic 2 is the sum of arc frequencies. (Recall that these are obtained by counting the number of times the corresponding substring matches between the input and the entire dictionary.) The scoring heuristics are one obvious dimension on which different versions of PbA can vary. In our work to date, we have followed D&N in giving primacy to Heuristic 1. The set of shortest paths (i.e., the candidate pronunciations) is found by a simple breadth-first search. Various possibilities exist for Heuristic 2. D&N took the sum of the arc frequencies along the path but Damper and Eastmond (1997) showed that the product of the arc frequencies worked better, and taking the sum of products over the multiple paths for identical pronunciations improved performance even more. Our most successful approach, however, has been to use multiple scoring strategies which are then combined to reach a final decision, as described in the following subsection.

Note that this particular implementation of PbA does not guarantee an output pronunciation in all cases. A complete path through the lattice requires that all nodes on that path (except the first and last) have at least one arc incident on the node and at least one arc leaving it. (In the terminology of graph theory, both the indegree and outdegree of each node must be greater than or equal to one.) Clearly, each arc must have a node at either end. Although an arc may have an empty label, a node cannot. Hence, the minimum matching segment length corresponds to a letter bigram. It is possible that no matching bigram exists in some cases. Consequently, there is no complete path through the lattice and no pronunciation can be inferred. (That is, the inference method is not *complete*.) This is the *silence problem*. It can be avoided by using a different form of lattice (e.g., Sullivan and Damper 1993) in which the nodes represent spaces, or *junctures*, between letters rather than the letters themselves, but at the cost of considerably more computation. At much less expense, Damper and Marchand (1998) used the following simple heuristic to good effect: If there is no complete path through the lattice but a length-1 gap at some point that would complete the path, then fill this gap with a unit-weight arc.

2.4 Limitations and improvements

With the above description as background, it is apparent that PRONOUNCE has some underlying assumptions and possible limitations that should be made explicit. Like any data-driven method, it can only be as good as its data – the lexicon. This has to be assumed to be both correct and representative. Further, there is no sound theoretical basis for the two heuristics that it uses. Although it seems reasonable to favour longest substring matches as well as commonly-occurring matches, there is no guarantee that this is an optimal or near-optimal strategy. As we have seen, there is not even a guarantee of an output in all cases (the silence problem). It has poor worst-case computational complexity, as detailed in the next subsection, but its average complexity seems to be more than acceptable for practical deployment. In spite of any apparent limitations, our previous experiences of the approach convince us that it is well worth the investment of serious, further study.

Our current version of PbA (Marchand and Damper 2000) features several amendments to the D&N formulation, which seem to have a generally beneficial impact on performance. First, we use full pattern matching between input letter string and dictionary entries, as opposed to D&N's partial matching, which was described above. This considers all possible overlaps in finding matching substrings. Thus, rather than starting with the two strings left-aligned, we start with the initial letter of the input string $\mathscr{I}$ aligned with the end letter of the dictionary entry $\mathscr{D}$. The matching process terminates not when the two strings are right-aligned, but when the end letter of $\mathscr{I}$ aligns with the initial letter of $\mathscr{D}$.

Second, although we retain D&N's Heuristic 1 unaltered, we use a quite different version of Heuristic 2, whereby multiple (five) heuristics are used to score candidate pronunciations corresponding to non-unique shortest paths. These rank order the candidates according to:

1. the maximum product of the arc frequencies (*PF*) along the shortest path;
2. the minimum standard deviation of the arc lengths along the shortest path (called *SDPS* by Marchand and Damper);
3. the maximum frequency of the same pronunciation (*FSP*) within the shortest paths;
4. the minimum number of different symbols (*NDS*) between a pronunciation and the other candidates;
5. the maximum weak link (*WL*) value, where the weak link is the minimum of the arc frequencies.

These strategies are used to rank order the candidates, and a fixed number of points is distributed among the candidates according to their position in the ranking. Individual points are then multiplied together to produce a final overall score and the best-scoring pronunciation is selected as the output. In recent work, Damper and Marchand (forthcoming) show that this rank fusion approach gives statistically significant performance improvements not only over simpler versions of PbA, but over the several other fusion schemes that were tried.

2.5 Computational complexity of PbA

If PbA is to be of practical use in TTS systems, then it must be computationally efficient. A reasonable measure of complexity is the number of distinct paths in the lattice, since this will affect both the memory requirements (to store the lattice, the search tree of candidate paths, etc.) and the time requirements (to search for the set of shortest paths, to disambiguate the paths using multiple strategies, etc.). It should be clear that this number is potentially very large. Let l be the length of the longest word in letters. In fact, $l = 19$ in this work. In the absolute worst case that every letter associates with every possible phoneme, all pattern matchings are bigrams (i.e., there are no arcs longer than length-1) and supposing there are 50 phonemes in the inventory, the number of distinct paths is astronomical at $19^{50} \sim 10^{63}$. Hence, it is important for our method that the average complexity is well below the worst case. Fortunately, this turns out to be the case (see last paragraph of section 5.1).

3 Lexical database

The lexical database on which this work is based is the 20,009 words of *Webster's Pocket Dictionary*, as used by Sejnowski and Rosenberg (1987), hereafter S&R, for training their NETtalk neural network. The database is publicly available via the World Wide Web from:

`ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries/`
(last accessed 8 April 2004)

For consistency with our previous work, homonyms (413 entries) were removed from the original NETtalk dataset to leave 19,596 entries. We exclude from our definition of homonym those word pairs which are spelled alike and pronounced alike but are different in meaning, such as <bank> (financial institution) and <bank> (in the sense of a bank of earth). These are considered to be the same word, and do not appear separately in the database anyway, which omits any definition of meaning.

S&R have manually aligned the data, to impose a strict one-to-one correspondence between letters and phonemes – see Damper, Marchand, Marsters and Bazin (2005) for extensive discussion of this alignment process and an algorithm for doing it automatically. They have also indicated stress and syllabification patterns for each word. The form of the data (rearranged in columns for ease of presentation) is:

aardvark	`a-rdvark`	1 <<<> 2 <<	
aback	`xb@k-`	0 > 1 <<	
abacus	`@bxkxs`	1 < 0 > 0 <	
abaft	`xb@ft`	0 > 1 <<	etc.

The second column is the pronunciation, expressed in the phoneme symbols listed in S&R's Appendix A, pp. 161–162. In this paper, we retain the use of S&R's phonetic symbols rather than transliterating to the symbols recommended by the International Phonetic Association. We do so to maintain consistency with S&R's

publicly-available lexicon. The '–' symbol is the null phoneme, introduced to give a strict one-to-one alignment between letters and phonemes to satisfy the training requirements of NETtalk. The phoneme inventory is of size 51, including the null phoneme and additional phonemes (/K/, /X/ and /#/) invented to avoid the use of null letters when one letter corresponds to two phonemes, as in <x> → /ks/.

The third column encodes the syllable boundaries for the words and their corresponding stress patterns. According to S&R (Appendix A):

<	denotes	syllable boundary (right)
>	"	syllable boundary (left)
1	"	primary stress
2	"	secondary stress
0	"	tertiary stress

Stress is associated with vowel letters and arrows with consonants. The arrows point towards the stress nuclei and change direction at syllable boundaries. This information is not adequate by itself to place syllable boundaries directly. We can, however, infer four rules (or regular expressions) to identify syllable boundaries. Denoting boundaries by '|':

R1:	[<>] ⇒ [< | >]
R2:	[< digit] ⇒ [< | digit]
R3:	[digit >] ⇒ [digit | >]
R4:	[digit digit] ⇒ [digit | digit]

These have been confirmed as correct by Sejnowski (personal communication). For the above four example words, the syllabifications are <aard | vark>, <a | back>, <ab | a | cus> and <a | baft>.

The longest word in the dictionary is <counterintelligence>. Figure 3 shows the distribution of the number of syllables in Webster's dictionary. The majority of words has 2 syllables; just one word has 8 syllables, namely <ir | rec | on | ci | la | bil | it | y> (2 <> 2 < 0 <> 2 > 0 > 1 < 0 < 0). Excluding implicit word start and end delimiters, there are 124,771 junctures between letters, i.e., 124,771 possible placements of syllable boundaries. Of these possibilities, 30,419 (or 24.38%) are actual syllable boundaries.

4 Syllabification by analogy

One of the outstanding advantages of lazy learning approaches to language processing is the ease with which algorithms can be transferred to new tasks. PbA is no exception; the basic algorithm is readily modified to perform syllabification by analogy. All that is needed is an evidence base of already-syllabified examples (but always bearing in mind the need to be sceptical about its status as a gold standard). The evidence base was prepared for the 19,596 words of the S&R version of Webster's dictionary using the regular expressions R1–R4 of the previous section.

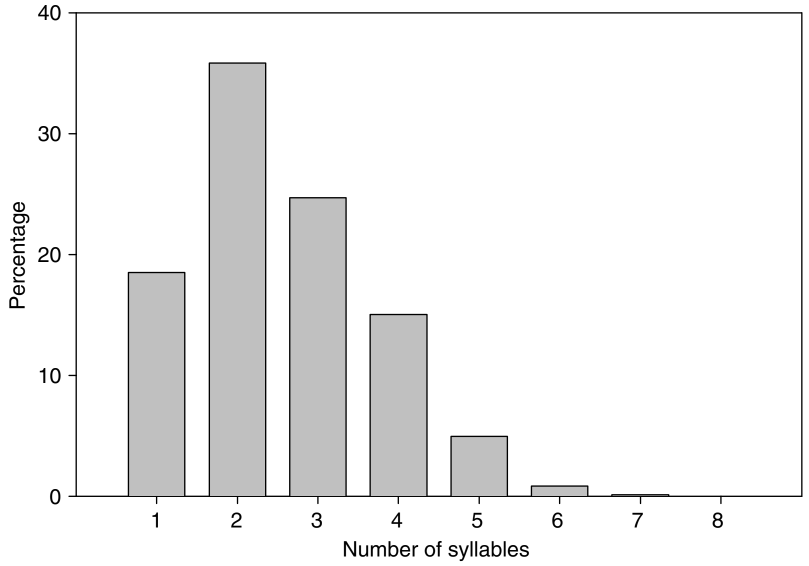

Fig. 3. Distribution of the number of syllables per word in Webster's dictionary.

4.1 Principles

Syllabification by analogy closely follows the principles of PbA set out in section 2. The major modification (for syllabification in the orthographic domain) is to represent all junctures between letters explicitly. This representation has to be different in the case of:

1. input words, where the syllabification is unknown;
2. lexical entries, where it is known;
3. the SbA output, where it is inferred.

For example, the input word <abbey> is expanded to <a * b * b * e * y>. Here the '*' symbol merely indicates the *possibility* of a syllable boundary. On the other hand, a dictionary entry such as <ab | nor | mal> is expanded to <a * b | n * o * r | m * a * l>. In this case, the '*' symbols indicate the known absence of a syllable boundary. During pattern matching, '*' in the input is allowed to match either with '*' or with '|' in the dictionary entries. A '* – *' match is entered into the syllabification lattice as a '*' whereas a '* – |' match is entered into the syllabification lattice as a '|'. The syllabification lattice has exactly the same form as the pronunciation lattice, except that '*' is explicitly represented as an input symbol (labelling nodes), '*' and '|' are explicitly represented as possible output symbols (labelling arcs), and there is no pronunciation information labelling the nodes and arcs. From here, the process proceeds exactly as for PbA, fully described in section 2, eventually producing as output a syllabified version of <abbey> such as <a * b | b * e * y>, from which the '*' symbols are removed to yield the final output <ab | bey>. The necessary modifications to perform syllabification in the pronunciation domain should now be obvious to the reader.

Although the required modifications to PbA are minimal, we do approximately double the size of the strings with which we are dealing. In fact, for a word of length l, we increase the possible number of arcs in the lattice by a factor of 2^{l-1} by introducing '$*$' in the input representation, which can associate with either '$*$' or '$|$' in the pronunciation at $l - 1$ junctures. This leads to a significant increase in run times for SbA relative to PbA.

4.2 Results of SbA

As discussed earlier, our standard PbA model (Marchand and Damper 2000) uses five different scoring strategies for Heuristic 2. The various combinations of strategy are denoted as a 5-bit code where '1' at position i indicates that strategy i was included in the combination and a '0' indicates that it was not. Thus, as an example, the code 00010 indicates that Strategy 4 (*NDS*) was used alone. We will use the same convention in presenting results for syllabification.

Table 1 shows the percentage of words correctly syllabified by SbA for all 31 possible combinations of strategy. To be considered correct, *all* boundaries in a word (i.e., both '$*$' and '$|$') must be correctly identified. We also give the overall percentage of boundaries correctly identified as '$*$' and '$|$'. Since words correct is the more demanding measure (Damper *et al.* 1999, Müller 2002), we use this to determine what is considered to be the best result, using the leave-one-out methodology.

On this basis, our best result (shown in bold in Table 1) is 78.10% words correct, for the 10101 combination. In this case, 93.1% of boundaries (junctures) between letters are correctly identified either as syllable boundaries or as non-syllable boundaries. We also show underlined the word accuracy results for the three strategies contributing to this best result. These are Strategy 3 (*FSP*, 77.08%), Strategy 5 (*WL*, 70.01%) and Strategy 1 (*PF*, 66.04%). Note that Strategy 1 by itself did worse than Strategy 4 (*NDS*, 69.71%) yet Strategy 1 participates in the best combination whereas Strategy 4 does not. Considering the best figure of 78.10% words correct versus the best single strategy result of 77.08%, the improvement may look superficially marginal. However, the very large numbers of words tested (tens of thousands) make this improvement highly significant (binomial test, one-tailed, $z = 3.406$, $p < 0.001$).

The results indicate that Strategy 3, *FSP*, is consistently well performing with over 75% words correctly syllabified whenever it is employed in a combination. This parallels its consistently good performance when used for PbA, as discussed extensively in Marchand and Damper (2000), Damper and Marchand (forthcoming) and Damper *et al.* (2005).

Unfortunately, in the absence of a universally-agreed set of canonically correct syllabifications, it is hard to determine the quality of these results. Subjectively, 93.1% boundaries correct appears to be a very creditable performance for a language like English, but we urgently need some objective reference against which to assess this figure. We have in fact implemented a range of other syllabification algorithms and compared them to SbA, but space does not allow us to report the results here. They will be the subject of a future publication. Even with a careful comparison such

Table 1. *Results of SbA*

Combination	Words correctly syllabified (%)	Junctures correctly identified (%)
00001	70.01	91.29
00010	69.71	92.13
00011	72.19	92.60
00100	77.08	92.80
00101	77.98	93.12
00110	75.43	92.56
00111	76.38	92.86
01000	55.51	87.29
01001	67.34	90.51
01010	70.30	92.15
01011	72.41	92.52
01100	76.09	92.58
01101	77.23	92.92
01110	75.12	92.56
01111	76.20	92.84
10000	66.04	90.27
10001	70.29	91.26
10010	72.19	92.36
10011	74.08	92.73
10100	77.20	92.82
10101	**78.10**	93.10
10110	75.97	92.69
10111	76.96	92.99
11000	69.76	91.07
11001	70.22	91.26
11010	73.17	92.58
11011	74.04	92.73
11100	77.54	92.99
11101	77.49	93.03
11110	76.40	92.88
11111	76.94	93.05

as we have done, in the absence of a gold standard, it remains difficult to determine which is the *best* of the compared algorithms. We can only say which performed best relative to the evidence base(s) used. So for the purposes of this paper, we decided that the potential and ability to improve pronunciation by analogy should be the target test of SbA.

5 Inferring syllabification and pronunciation together

Throughout, a major motivation for this work has been that knowledge of syllable boundaries might improve pronunciation by analogy and, thereby, the overall performance of TTS systems. In this section, we describe and present results for three different algorithms designed to test this possibility.

5.1 Three models of syllabification/pronunciation

These are:

1. S*(P)bA, which assumes that perfect (according to the evidence base) orthographic syllabification has already been achieved;
2. a so-called *parallel model*, denoted (S||P)bA, in which syllabification and pronunciation are simultaneously inferred from orthographic input, as a single process;
3. a so-called *series model*, denoted (S+P)bA, in which syllabification is first inferred in the orthographic domain using SbA and then pronunciation is inferred using a form of PbA extended to cater for syllable boundaries in the input.

Note that the brackets in the notation for these three algorithms surround the information that is inferred by analogy.

The *perfect model*, S*(P)bA requires that PbA is straightforwardly extended to cater for syllable boundaries in the orthographic input. In this case, the input is the syllabified spelling pattern of the word to be pronounced, e.g., <abbey> would be input to S*(P)bA in the form <ab | bey>. This input string is then compared to lexical entries <aard | vark>, <a | back>, <ab | a | cus>, <a | baft>, . . . , <ab | bess>, etc. giving rise to substring matches such as < | b>, <ab | > and <ab | b>. (Note that for clarity of explanation we have ignored word boundaries, i.e., '#' matching *Start* and *End* nodes.) These substrings are entered, together with their corresponding pronunciations (including syllable markers), into the pronunciation lattice in the usual way. From this point, the process proceeds just as for normal PbA, as described in section 2.

In the parallel model, (S||P)bA, there are of course no syllable markers ('|') in the input, as the syllabification is unknown. The process is very like that of SbA, except that in this case we include pronunciation information when building the syllabification/pronunciation lattice. As for SbA, an input word such as <abbey> is expanded to <a * b * b * e * y> and this is compared to lexical entries in the form <a * a * r * d | v * a * r * k>, <a | b * a * c * k>, <a * b | a | c * u * s>, <a | b * a * f * t>, . . . , <a * b | b * e * s * s>, etc. Continuing as for SbA, ' * ' in the input is now allowed to match either with ' * ' or with ' | ' in the dictionary entries and a ' * – * ' match is entered into the lattice as a ' * ' whereas a ' * – | ' match is entered into the lattice as a ' | '. Again ignoring word boundaries for simplicity, this will give rise to lattice entries such as <a | b * >, <a * b | > and <a * b | b>, together with their corresponding pronunciation substrings (including syllable markers). As before, the process now continues as for normal PbA, yielding a syllabified pronunciation such as /@b | – i– / from which the syllable boundaries are removed for scoring.

The series model, (S+P)bA, simply uses SbA as described in section 4 to syllabify the input. From this point, processing is just as for the S*(P)bA model although, in this case, the orthographic syllabification is of course not 'perfect'.

By the introduction of juncture symbols ' * ', we are increasing the size of the input and thereby the complexity of the parallel and series algorithms relative to PbA. Since the series model is but a small extension to SbA, its complexity is not much greater than SbA, which we already know (from computation of the results in section 4) to run in reasonable time. We were, however, concerned about the more complex parallel model, (S||P)bA. Accordingly, for this algorithm, we excluded from further processing any word for which the number of candidate pronunciations exceeded 50,000. In the event, only one word (<caoutchouc>) out of 19,596 failed this test. It had 122,316 candidate pronunciations, all of shortest length 5. This is obviously a highly unusual loan (from French) which few English speakers would recognise or accept as a word of their language. The word with the next largest number of candidates was <eisteddfod>, another loan word (from Welsh) with 41,208 length-5 shortest paths.

5.2 Results for syllabification/pronunciation

For compactness, for each of the three algorithms, we tabulate results for the best single strategy, the best combination overall, and for all five strategies used together (11111). By 'best' in this context, we refer to word accuracy since this is a more demanding measure than symbol (phoneme and/or juncture) accuracy (Damper *et al.* 1999, Müller 2002). For a word to count as a correct pronunciation, *all* phonemes of a word must be correct, including the null phoneme, evaluated on a one-to-one basis. Thus, if the correct pronunciation for <make> is /meIK-/, then /meIK-/ is scored as incorrect. This is a deliberately strict way of scoring. As regards phoneme accuracy, this is again assessed on a one-to-one basis counting nulls as legitimate symbols. Hence, /meIK-/ scores just 20% phonemes correct as a pronunciation of <make>. Similarly, for a word syllabification to count as correct, all junctures must be classified correctly as either a syllable boundary ('|') or as a non-syllable boundary ('∗'). Some consideration was given to removing the null phonemes before evaluation, as they are really just a left-over from the alignment process. We found, however, that this made almost no difference to the results. For comparability with our earlier papers, we decided to leave nulls as part of the evaluation.

5.2.1 Results of $S^*(P)bA$

Table 2 shows the results of S*(P)bA compared to those for PbA without syllabification. The improvement from 65.35% words correct for PbA to 71.74% words correct for S*(P)bA is dramatic. Using a binomial test, we obtain $z = 19.86$ corresponding to a probability level that is far too small to compute. An improvement of this magnitude demonstrates amply the benefit that syllabification can bring to the performance of pronunciation systems if it is done properly.

The task now is to reproduce gains in pronunciation performance of this kind when the syllabification of the input has to be automatically inferred. This is a real challenge, because automatic syllabification is difficult (see section 1) and getting the syllabification wrong is potentially very disruptive. Since pattern matching is exact (substrings either match perfectly or not at all), an incorrect boundary symbol will prevent substrings matching when they should.

Table 2. *Results of $S^*(P)bA$ assuming correctly syllabified input versus results for PbA without syllabification. A very significant performance gain is achievable if syllabification can be inferred with high accuracy*

	Best Single	Best Combination	All Combinations
PbA	00100	11111	11111
Words (%)	62.91	65.35	65.35
Phonemes (%)	91.63	92.40	92.40
S^*(P)bA	00100	10100	11111
Words (%)	69.15	71.74	71.08
Phonemes (%)	93.57	94.12	94.05

Table 3. *Results of the parallel (S||P)bA model in which syllabification and pronunciation are simultaneously inferred by analogy*

	Best Single	Best Combination	All Combinations
(S||P)bA syllabification	00100	00101	11111
Words (%)	76.65	77.71	76.89
Junctures (%)	92.76	93.14	93.24
(S||P)bA pronunciation	00100	10100	11111
Words (%)	62.74	64.82	62.79
Phonemes (%)	91.67	92.25	92.21

5.2.2 Results for (S||P)bA

Table 3 shows the obtained syllabification and pronunciation performance for the parallel model. Comparing Table 3 with Table 1 shows that the parallel model fails to deliver any improvement in syllabification over SbA. Here we obtain a best result of 77.71% words correctly syllabified for combination 00101 versus 78.10% for SbA; i.e., (S||P)bA is marginally poorer ($p \sim 0.09$).

Comparing Table 3 with Table 2, the parallel model also failed to deliver any pronunciation improvement over PbA without syllabification. Here we obtain a best result of 64.82% words correctly pronounced for combination 10100 versus 65.35% for PbA; i.e., (S||P)bA is marginally poorer ($p \sim 0.06$).

5.2.3 Results for (S+P)bA

Table 4 shows pronunciation results obtained from the series model in which SbA is followed by the inference of pronunciation. (Syllabification results are not shown separately because they are the same as for SbA in Table 1.)

The results are again disappointing. They are slightly inferior to the (S||P)bA results ($p \sim 0.05$) although obtained with much less computation. This seems to confirm the emerging picture that if syllabification is done well, it can contribute to improved pronunciation, but errors in syllabification can be quite harmful to the process.

Table 4. *Results of the series (S+P)bA model in which pronunciation is inferred after syllabification by SbA*

	Best Single	Best Combination	All Combinations
(S+P)bA pronunciation	00100	10100	11111
Words (%)	62.20	64.26	63.66
Phonemes (%)	91.22	91.68	91.67

5.3 Further analysis of results

First, in view of our earlier remarks about the average computational complexity, we note that the average number of arcs in the pronunciation lattice and the average number of candidate pronunciations were both very reasonable, leading to conveniently short run times. For PbA, S^*(P)bA and (S+P)bA, the average number of arcs increased from approximately 70 for monosyllabic words to about 370 for words of 6 syllables, while the number of candidate pronunciations rose from an average of about 5 to about 20. (S||P)bA produced much larger lattices, with monosyllabic words averaging 450 arcs and 35 candidate pronunciations, and 6-syllable words averaging 2600 arcs and 160 candidates. Consequently, (S||P)bA was considerably more time consuming to run. As a matter of interest, the order of run times on a Dell Optiplex GX270, Pentium 4CPU 2.40 GHz personal computer was as follows: The perfect model ran overnight, the series model took just over two days and the parallel model was much slower (taking several days).

Table 5 shows the transcription accuracy by individual letter of the alphabet for PbA and the three versions of syllabification-with-pronunciation studied in this paper, together with the percentage which each letter contributes to the dictionary. These results confirm the well known phenomenon for English that vowel letters are much harder to transcribe than consonants, and they also constitute a large part ($\sim$ 40%) of the dictionary. For PbA, the individual accuracies for letters <a>, <e>, <i>, <o> and <u> are all below 90% whereas for the remaining letters they are all above 90%. The easiest letter to transcribe is <v>, with greater than 99.4% accuracy for all four methods. The hardest letter to transcribe is <o>, with overall poorest accuracy of 82.58%. As we wrote in an earlier paper (Marchand and Damper 2000, p. 216): ‘Future work should therefore attempt to find scoring methods and combination techniques that deal effectively with the vowels.’ S^*(P)bA apparently does just this, showing large gains on all the vowel letters but having little impact on the transcription of consonant letters. But unfortunately, neither (S||P)bA nor (S+P)bA are able to replicate S^*(P)bA’s success on the vowel letters.

Figure 4 shows pronunciation performance for PbA and the three syllabification-with-pronunciation algorithms as a function of the number of syllables in each word. Words of 7 and 8 syllables have been removed from the plot as there were so few of them (24 and 1 respectively) that these data points were effectively meaningless.

For monosyllabic words, the performance of all algorithms except (S||P)bA was essentially identical to that of PbA. This is only to be expected for S^*(P)bA: If there are no syllable boundaries, then they cannot help to infer pronunciation.

Table 5. *Percentage accuracy in transcribing individual letters of PbA and the three versions of syllabification-with-pronunciation studied in this paper. All figures are for all combinations of the five scoring strategies, 11111*

Letter	% in Webster's	PbA	S*(P)bA	(S||P)bA	(S+P)bA
<a>	8.99	82.67	87.81	82.27	81.92
<b>	2.07	98.86	98.49	99.20	98.13
<c>	4.65	96.93	95.99	96.16	95.85
<d>	2.94	98.35	98.40	98.64	97.88
<e>	10.95	85.78	91.15	85.70	85.81
<f>	1.39	98.95	97.70	99.60	97.55
<g>	2.15	94.56	94.33	93.66	93.72
<h>	2.36	97.77	98.56	97.71	97.07
<i>	8.83	88.21	92.05	88.40	87.79
<j>	0.19	94.49	95.59	96.69	95.22
<k>	0.78	97.86	97.78	98.13	96.80
<l>	5.53	96.13	96.78	96.48	94.83
<m>	3.20	99.50	98.70	99.57	98.64
<n>	6.82	97.10	97.08	96.81	96.02
<o>	6.87	82.58	87.47	81.87	82.25
<p>	3.21	99.48	98.79	99.48	98.66
<q>	0.24	98.25	99.13	99.13	97.96
<r>	7.46	96.05	96.22	96.05	95.27
<s>	5.29	96.08	94.47	95.11	94.07
<t>	7.65	98.08	97.94	97.89	97.01
<u>	3.83	86.01	87.43	85.08	85.59
<v>	1.23	99.60	99.60	99.83	99.43
<w>	0.85	97.32	98.13	97.81	97.08
<x>	0.31	90.83	94.77	93.06	91.95
<y>	1.84	93.87	94.77	93.31	93.76
<z>	0.36	95.53	92.82	95.92	94.37

The relative poor performance of (S||P)bA indicates that it erroneously interpolates syllable boundaries in monosyllabic words; (S+P)bA seems not to do this. Otherwise, performance is generally quite similar for PbA, (S||P)bA and (S+P)bA and is significantly poorer than S*(P)bA. An exception is that (S+P)bA does less well that the other algorithms on words of 6 syllables.

Since we adopt D&N's Heuristic 1 throughout, giving primacy to shortest paths through the pronunciation lattice, it is worth investigating the limits that this sets on the attainable performance. In Table 6, we show the minimum and maximum possible words correct percentage for PbA and for the three syllabification-with-pronunciation algorithms studied here. The minimum obtains when all shortest paths through the lattice correspond to the same correct pronunciation, so that Heuristic 2 (whether using multiple strategies or otherwise) becomes irrelevant. The maximum obtains when the correct pronunciation is among the set of shortest paths, i.e., the assumption is that Heuristic 2 that selects among the candidates does so perfectly. It might also be interesting to know how many times the correct pronunciation fails

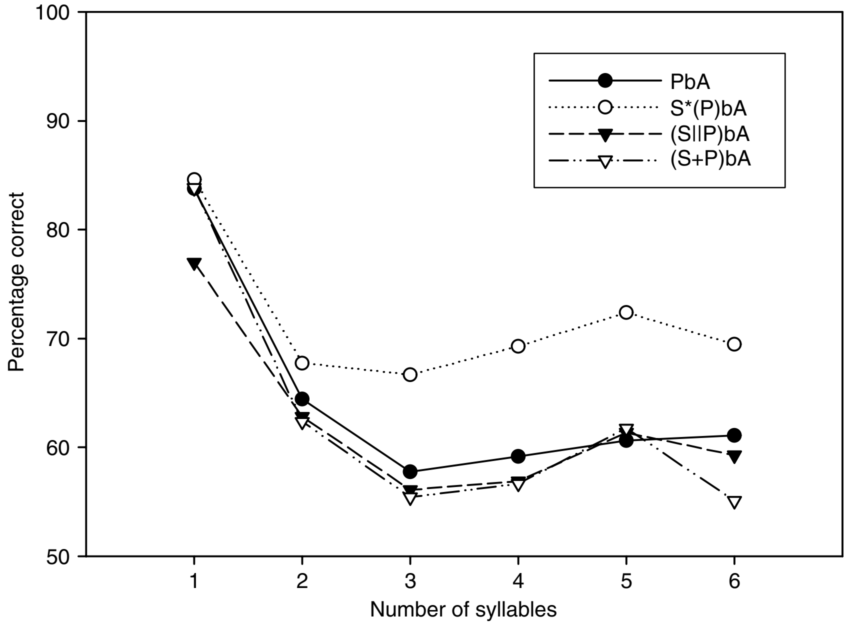

Fig. 4. Pronunciation performance as a function of the number of syllables in a word.

Table 6. *Minimum and maximum pronunciation performance (words correct) set by tacit acceptance of the shortest-path heuristic. These are compared to the actual performance obtained by PbA and the three syllabification-with-pronunciation algorithms studied in this paper*

	PbA	S*(P)bA	(S||P)bA	(S+P)bA
Min. (%)	15.33	16.30	2.55	14.91
Max. (%)	85.12	88.09	90.23	79.11
Actual best (%)	65.35	71.74	64.82	64.26

to appear in *any* path, but we do not yet know this because in its present state our algorithm does not compute all possible paths through the lattice, only shortest paths since the latter gives an enormous saving in computer time.

Table 6 shows that syllabification can potentially improve the chances that the correct pronunciation is found among the set of shortest paths, i.e., it improves the maximum from the PbA baseline in two of the three cases, the exception being (S+P)bA. For the parallel model, however, the minimum is reduced quite dramatically to just 2.55%. That is, the simultaneous inference of syllabification and pronunciation appears to introduce greater variability into the set of shortest paths, and this seems to be at least in part behind the disappointing performance of this model. Perhaps the most telling conclusion, however, is that although syllabification has the potential to improve pronunciation by analogy, the challenge remains to find

Table 7. *Performance of PbA, (S||P)bA and (S+P)bA separately assessed on words that SBA syllabified correctly and those that SbA did not syllabify correctly*

	PbA	(S||P)bA	(S+P)bA
Good syllabification by SbA (76.94%)	74.07	72.67	76.08
Bad syllabification by SbA (23.06%)	36.25	29.83	22.24

better algorithms for inferring good syllabifications and with enhanced capability to select the correct pronunciation from the set of candidates.

Finally, we analysed how the three different practical pronunciation algorithms, PbA, (S||P)bA and (S+P)bA perform on two classes of words: (i) those correctly syllabified by SbA; (i) words incorrectly syllabified by SbA. The results are shown in Table 7, and all figures are for the 11111 combination of strategies. Consider first the results for PbA. In this case, since syllabification plays no part in the operation of the algorithm, we might well expect that performance will be even across the two classes of words. On the contrary, there is a dramatic difference: Those words that are correctly syllabified are pronounced with much higher accuracy (74.07%) than those words which are incorrectly syllabified (36.25%).

This is an intriguing finding. It implies that those words that are hard for SbA to syllabify are also hard to derive a correct pronunciation for automatically. (It could also suggest that, for some reason, such words were difficult for S&R to syllabify manually.) Whatever the precise relationship, it is clear that there is some interaction between ease of syllabification (either inferred automatically or done manually) and ease of pronunciation generation. The same pattern holds across (S||P)bA and (S+P)bA, where much higher accuracy is achieved on the properly syllabified words. This is perhaps less remarkable, as syllabification plays a part in pronunciation generation in both models. Nonetheless, the figures give a feel for the sort of performance gains that could be achieved if syllabification is done well and, conversely, the penalties of getting syllabification wrong.

6 Discussion and conclusions

Although controversy surrounds the status of the syllable as a unit in theoretical linguistics, the concept has considerable practical value in computational linguistics and speech technology. This paper shows conclusively that including good quality information on the syllabification of words can enhance the performance of a pronunciation system for use in text-to-speech and similar applications. However, this was done by including ‘perfect’ syllabification in the orthographic input to a pronunciation by analogy (PbA) system. By ‘perfect’, we mean that knowledge was assumed of the syllabification specified in the evidence base (i.e., a dictionary of word spellings, syllabifications and pronunciations) used for analogical reasoning. Such knowledge would not, of course, be available to any practical system.

The challenge is to secure similar improvements when the syllabification has to be inferred, as is the case in practice. We have presented two different systems that use analogical reasoning to produce both syllabifications and pronunciation. We call these the parallel and series models, (S||P)bA and (S+P)bA respectively. Unfortunately, neither improves on PbA without syllabification. The main reason for this seems to be that errors in automatically-inferred syllabification disrupt the pattern matching process within PbA, which works by finding perfect matches and is unable to tolerate errorful input. One possibility for future work is therefore to relax the strict matching criterion for junctures and syllable boundaries, and to allow approximate matches in building the syllabification/pronunciation lattice.

Because lazy learning techniques are easily exported to new related tasks, it was straightforward for us to modify pronunciation by analogy (PbA) to syllabification by analogy (SbA). At this stage, we cannot claim that SbA is competitive with other syllabification algorithms in the literature, because of the considerable difficulties of comparison. Competitor algorithms can, of course, be evaluated using the same performance measures on the same test data, e.g., Sejnowski and Rosenberg's syllabification of Webster's dictionary as used in this work, and we have in fact done this in work to be separately reported, but then questions about the status and validity of the test data come into even sharper focus. In the absence of a gold-standard corpus of canonically-correct syllabifications, comparisons between performance data so obtained are uncertain.

One possible rejoinder to such concerns is that the S&R manually-syllabified data were at least good enough to improve pronunciation by analogy in the perfect case of the S^*(P)bA model. From this perspective, Sejnowski and Rosenberg cannot have done that bad a job. Another possibility we are actively pursuing is to build a corpus from those words on which different, authoritative sources agree about both pronunciation and syllabification. Such consistency or overlap increases our confidence that the corpus is of high quality and can be treated as a gold standard. And of course, the quality of the corpus is important not only for testing, but also because it forms the evidence base for analogical inference. Overall the search for better evidence bases and algorithms seems very worthwhile, motivated by the knowledge that a 'perfect' orthographic syllabification can improve pronunciation generation substantially, so a good syllabification algorithm should be well worth finding.

References

Aha, D. W. (1997) Lazy learning. *Artificial Intelligence Review* **11**(1–5), 7–10.

Aha, D. W., Kibler, D. and Albert, M. (1991) Instance-based learning algorithms. *Machine Learning* **6**(1), 37–66.

Baayen, H. (2001) *Word Frequency Distributions*. Kluwer Academic.

Blevins, J. (1995) The syllable in phonological theory. In: J. Goldsmith (editor), *Handbook of Phonological Theory*, pp. 206–244. Blackwell.

Bouma, G. (2003) Finite state methods for hyphenation. *Natural Language Engineering* **9**(1), 5–20.

Brill, E. (1995) Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics* **21**(4), 543–566.

Clements, G. N. (1988) The role of the sonority cycle in core syllabification. Working Papers of the Cornell Phonetics Laboratory, WPCPL No. 2, Research in Laboratory Phonology, Cornell University, Ithaca, NY.

Crystal, D. (1980) *A First Dictionary of Linguistics and Phonetics*. André Deutsch.

Daelemans, W. and van den Bosch, A. (1992) Generalisation performance of backpropagation learning on a syllabification task. In: M. F. J. Drossaers and A. Nijholt (editors), *TWLT3: Connectionism and Natural Language Processing*, Enschede, The Netherlands, pp. 27–37. Twente University.

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

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

Damper, R. I. and Marchand, Y (forthcoming) Information fusion approaches to the automatic pronunciation of print by analogy. *Information Fusion* (accepted).

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.

Damper, R. I., Marchand, Y., Marsters, J.-D. S. and Bazin, A. I. (2005) Aligning text and phonemes for speech technology applications using an EM-like algorithm. *International Journal of Speech Technology* **8**(2), 149–162.

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

Federici, S., Pirrelli, V. and Yvon, F. (1995) Advances in analogy-based learning: False friends and exceptional items in pronunciation by paradigm-driven analogy. *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.

Goslin, J. and Frauenfelder, U. H. (2000) A comparison of theoretical and human syllabification. *Language and Speech* **44**(4), 409–436.

Greenberg, S. (1999) Speaking in shorthand – A syllable-centric perspective for understanding pronunciation variation. *Speech Communication* **29**(2–4), 159–176.

Hoard, J. W. (1971) Aspiration, tenseness and syllabification in English. *Language* **47**, 133–140.

Kahn, D. (1976) *Syllable-Based Generalizations in English Phonology*. Bloomington, IN: Indiana University Linguistics Club.

Kessler, B. and Treiman, R. (1997) Syllable structure and the distribution of phonemes in English syllables. *Journal of Memory and Language* **37**(3), 295–311.

Kiraz, G. A. and Möbius, B. (1998) Multilingual syllabification using weighted finite-state transducers. *Proceedings of 3rd European Speech Communication Association (ESCA)/COCOSDA International Workshop on Speech Synthesis*, Jenolan Caves, Australia, pp. 71–76.

Kohler, K. J. (1966) Is the syllable a phonological universal? *Journal of Linguistics* **2**, 207–208.

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

Müller, K. (2001) Automatic detection of syllable boundaries combining the advantages of treebank and bracketed corpora training. *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France, pp. 402–409.

Müller, K. (2002) Evaluating syllabification: One category shared by many grammars. *Proceedings of the Workshop "Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems" at the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, Las Palmas, Canary Islands, Spain, pp. 37–43.

Müller, K., Möbius, B. and Prescher, D. (2000) Inducing probabilistic syllable classes using multivariate clustering. *Proceedings of 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, China, pp. 225–232.

Pinker, S. (1999) *Words and Rules: The Ingredients of Language*. Weidenfeld and Nicolson.

Pulgram, E. (1970) *Syllable, Word, Nexus, Cursus*. The Hague, The Netherlands: Mouton.

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

Selkirk, E. (1982) The syllable. In: H. van der Hulst and N. Smith (editors), *The Structure of Phonological Representations*, Volume 2, pp. 337–383. Foris.

Sullivan, K. P. H. (2001) Analogy, the corpus and pronunciation. In: R. I. Damper (editor), *Data-Driven Methods in Speech Synthesis*, pp. 45–70. Kluwer Academic.

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

Treiman, R. and Zukowski, A. (1990). Toward an understanding of English syllabification. *Journal of Memory and Language* **29**(1), 66–85.

van Santen, J. P. H., Shih, C., Möbius, B., Tzoukermann, E. and Tanenblatt, M. (1997) Multilingual duration modelling. *Proceedings of 5th European Conference on Speech Communication and Technology, Eurospeech'97*, Volume 5, Rhodes, Greece, pp. 2651–2654.

Weijters, A. (1991) A simple look-up procedure superior to NETtalk? *Proceedings of International Conference on Artificial Neural Networks (ICANN-91)*, Volume 2, Espoo, Finland, pp. 1645–1648.

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