Automatic Syllabification in English: A Comparison of Different Algorithms

Yannick Marchand[1,2], **Connie R. Adsett**[1,2,3], **Robert I. Damper**[1,4]

[1] *Institute for Biodiagnostics (Atlantic), National Research Council Canada*
[2] *Faculty of Computer Science, Dalhousie University, Canada*
[3] *Faculty of Computer Science, University of New Brunswick, Canada*
[4] *Information: Signals, Images, Systems (ISIS) Research Group, School of Electronics and Computer Science, University of Southampton, U.K.*

Key words

analogy

computational linguistics

corpus linguistics

rule-based systems

speech technology

syllabification

Abstract

Automatic syllabification of words is challenging, not least because the syllable is not easy to define precisely. Consequently, no accepted standard algorithm for automatic syllabification exists. There are two broad approaches: rule-based and data-driven. The rule-based method effectively embodies some theoretical position regarding the syllable, whereas the data-driven paradigm tries to infer "new" syllabifications from examples assumed to be correctly syllabified already. This article compares the performance of several variants of the two basic approaches. Given the problems of definition, it is difficult to determine a correct syllabification in all cases and so to establish the quality of the "gold standard" corpus used either to evaluate quantitatively the output of an automatic algorithm or as the example-set on which data-driven methods crucially depend. Thus, we look for consensus in the entries in multiple lexical databases of pre-syllabified words. In this work, we have used two independent lexicons, and extracted from them the same 18,016 words with their corresponding (possibly different) syllabifications. We have also created a third lexicon corresponding to the 13,594 words that share the same syllabifications in these two sources. As well as two rule-based approaches (Hammond's and Fisher's implementation of Kahn's), three data-driven techniques are evaluated: a look-up procedure, an exemplar-based generalization technique, and syllabification by analogy (SbA). The results on the three databases show consistent and robust patterns. First, the data-driven techniques outperform the rule-based systems in word and juncture accuracies by a very significant margin but require training data and are slower. Second, syllabification in the pronunciation domain is easier than in the spelling domain. Finally, best results are consistently obtained with SbA.

Address for correspondence. Robert I. Damper, Information: Signals, Images, Systems (ISIS) Research Group, School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.; < rid@ecs.soton.ac.uk>

Language and Speech 0023-8309; Vol 52(1): 01–27; 099881; DOI:10.1177/0023830908099881
http://las.sagepub.com

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 centers on the difficulty of defining the syllable. Crystal (1980, p.342), for instance, 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 task." There is general agreement that a syllable consists of a *nucleus* that 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. Good general accounts of the controversy are provided by Treiman and Zukowski (1990) and Goslin and Frauenfelder (2001), with the former more specifically considering English—the language of interest in this article—and the latter focusing on French.

However it is defined, and whatever the rights or wrongs of theorizing about its linguistic status, native speakers of a language are easily able to count the syllables of a word based entirely on intuition. And if the syllable does act as a structuring device in composing words, as many believe, then knowledge of this structure could well aid word modeling in automatic speech recognition and/or the unit selection and composition process of concatenative synthesis. For instance, Müller, Möbius, and Prescher (2000, p.225) write "syllable structure represents valuable information for pronunciation systems." That is, the pronunciation of a phoneme can depend upon where it is in a syllable. In some quarters, there is thought to be a morphological influence on syllable structure, such that syllabification can predict good points at which to hyphenate words at line endings in computer typesetting software such as LATEX (Liang, 1983). So, arguments about the theoretical status of the syllable as a linguistic unit notwithstanding, there are good practical reasons for seeking powerful algorithms to syllabify words.

Traditional approaches to automatic syllabification have been *rule-based* (or knowledge-based), implementing notions such as the maximal onset principle (Kahn, 1976; Pulgram, 1970) and sonority hierarchy (Clements, 1988), including ideas about what constitute phonotactically legal sequences in the coda,[1] for instance. 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 competing hypotheses. For instance, Goslin and Frauenfelder (2001) document the disagreement between five different syllabification procedures on approximately 23,000 French words, with *agreement* assessed at "just over 58% of types" (p.420)—see below. Unfortunately, it is far from clear how the arbitration should be done, i.e., we do not actually possess the

[1] 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.

requisite knowledge to determine a canonically correct syllabification. As Goslin and Frauenfelder conclude: "the search for an answer to the problem of correct syllable segmentation is, as yet, unfinished" (pp.432–433).

Although syllabifications produced by naïve humans might provide a valid "gold standard" that could be potentially useful for arbitration, there are two major problems, as follows:

(1) Naïve humans do not necessarily agree in all cases. To cite Goslin and Frauenfelder (2001): "However, when listeners are asked to state explicitly where the syllable *boundaries* lie between … nuclei, great difficulties are encountered with differences of opinion arising between listeners (p.410). Syllabification is known to be influenced by familiarity of the subject with orthography (Treiman, Bowey, & Bourassa, 2002; Goslin & Floccia, 2007), leading to adult/child differences. A possible solution to this problem is to take majority, or "preferential" syllabifications, yet this necessarily weakness the status as a gold standard.

(2) It is impractical to collect large numbers (on the order of tens of thousands) of human syllabifications. For instance, Goslin and Frauenfelder (2001) perform human experimentation using just 138 tokens with only 57 intervocalic consonant singletons/cluster types in bisyllabic French non-words. This accounts for only 13% of the 431 possible consonant clusters in their lexical source (BDLEX). Furthermore, the context of intervocalic singletons/clusters was not considered; as the authors write: "One influence which has not been considered in this study is that of the vowel" (p.432). Clearly, it would be an imperfect gold standard that was based on these results, having such weak coverage of cases and ignoring vowels!

An alternative to the rule-based methodology is the *data-driven* (or corpus-based) approach, which attempts to infer "new" syllabifications from an evidence base of already-syllabified words (a dictionary or lexicon[2]), i.e., the corpus acts as the gold standard. Data-driven methods are therefore based on machine learning and, for the purposes of this article, it is convenient to draw a distinction between "lazy" and "eager" learning. By lazy learning (Aha, 1997; Aha, Kibler, & Albert, 1991), 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." A typical form of eager learning is the well-known back-propagation algorithm (Rumelhart, Hinton, & Williams, 1986) for training artificial neural networks, in which the training data are encoded into a small set of connection weights and thresholds. There is a small existing literature on data-driven syllabification. In particular, Daelemans and van den Bosch (1992) compare various methods for Dutch and show that the generalization performance of back-propagation learning is not better than symbolic (knowledge-based) approaches and that both are inferior to a form of lazy learning

[2] In this article, we will use the terms evidence base, lexical database, dictionary, corpus, and lexicon more or less interchangeably. Note, however, that we refer to the public-domain resources used here as "dictionaries" and reserve the term "database" for dictionaries that have been preprocessed, for example, by the removal of homonyms.

that they call "exemplar-based generalization." Kiraz and Möbius (1998) present a probabilistic syllabification algorithm in which observed frequencies of onsets, nuclei and codas are converted into the weights of a weighted finite state transducer. Müller et al. (2000) describe a hybrid (partly rule-based and partly data-driven) approach in which a form of the expectation-maximization (EM) algorithm is used to cluster example data in three- and five-dimensional syllable classes. The three-dimensional data are onset, nucleus and coda; the five-dimensional data add position of syllables in the word and stress type. The overall system uses a hand-crafted rule-based system (see Müller, 2001) that produces (all possible) candidate pronunciations; these are then ranked by the probabilistic syllable model and the most probable analysis selected to give the predicted pronunciation. Marchand and Damper (2007) describe a lazy learning procedure called syllabification by analogy (SbA) in which 78.1% of the approximately 20,000 words in *Webster's Pocket Dictionary* are correctly syllabified, but they do not assess how this performance compares to other approaches.

This article compares the performance of several variants of the two basic approaches to automatic syllabification: rule-based and data-driven. Our work attempts to be *predictive* and *empirical*, aimed at finding good syllabifications for practical application in speech technology and computational linguistics, rather than *descriptive* and *theoretical*, aimed at explaining experimental data and/or giving insight into any linguistic theory of the syllable. The remainder of this article is structured as follows. In the next section, we describe the three dictionaries used as the source of inferential data for the data-driven methods and against which all five methods were evaluated. We then describe the five automatic syllabification procedures: two rule-based and three data-driven. Next, we describe the results before discussing their implications for automatic syllabification, and concluding.

2 Electronic lexical databases

A key issue in assessing algorithms for automatic syllabification is the quality of the "gold standard" corpus used to define the correct result. Further, in the data-driven paradigm, this corpus forms the evidence base for inferring new syllabifications; hence, it is vital that its quality can be assured. This, however, is extremely difficult to do in the absence of any way of determining canonically correct syllabifications. Our approach is to use multiple dictionaries and to seek consensus among them, so as to reduce the possibility that our results are affected by the choice of a particular, idiosyncratic corpus.

In this work, we use two public-domain dictionaries, namely *Webster's Pocket Dictionary* and the *Wordsmyth English Dictionary-Thesaurus*, as the sources from which we derive three lexical databases, as now described.

2.1 *Webster's Pocket Dictionary*

The primary lexical database in this work is derived from *Webster's Pocket Dictionary* (20,009 words), as used by Sejnowski and Rosenberg (1987) to train their NETtalk neural network. The dictionary is publicly available for non-commercial use from ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries/ (last accessed 8 April 2004).

For consistency with our previous work on pronunciation using this dictionary, homonyms (413 entries) were removed from the original NETtalk dataset leaving 19,596 entries. Sejnowski and Rosenberg have manually aligned the data, to impose a strict one-to-one correspondence between letters and phonemes.[3] The phoneme inventory is of size 51, including the null phoneme and "new" phonemes (e.g., /K/ and /X/) invented to avoid the use of null letters when one letter corresponds to two phonemes, as in <x> → /ks/. The null phoneme (represented by the "–" symbol) was introduced to give a strict one-to-one alignment between letters and phonemes to satisfy the training requirements of NETtalk. In this article, we retain the use of the original phonetic symbols (see Sejnowski & Rosenberg, 1987, Appendix A, pp. 161–162) rather than transliterating to the symbols recommended by the International Phonetic Association. We do so to maintain consistency with this publicly-available lexicon.

In addition to the pronunciation, Sejnowski and Rosenberg have also indicated stress and syllabification patterns for each word. The form of the data is:

accumulate	xk-YmYlet-	0 < > 1 > 0 > 2 < <
adaptation	@d@pteS-xn	2 < 2 < > 1 > 0 < <
albatross	@lbxtrcs-	1 < > 0 > > 2 < < etc.

The second column is the pronunciation and the third column encodes the syllable boundaries for the words and their corresponding stress patterns:

<	denotes	syllable boundary (right)
>	denotes	syllable boundary (left)
1	denotes	primary stress
2	denotes	secondary stress
0	denotes	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. To this extent, "syllable boundary (right/left)" is a misnomer because 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]

[3] See Damper, Marchand, Marsters, and Bazin (2005) for extensive discussion of this alignment process and an algorithm for doing it automatically.

Table 1

Examples of stress and syllabification patterns

	Word		
	accumulate	*adaptation*	*albatross*
Stress pattern	0 < > 1 > 0 > 2 < <	2 < 2 < > 1 > 0 < <	1 < > 0 > > 2 < <
Syllabification	ac|cu|mu|late	ad|ap|ta|tion	al|ba|tross
Digit stress	00|11|00|2222	22|22|11|0000	11|00|22222

These have been confirmed as correct by Sejnowski (personal communication). Table 1 gives the syllable patterns of the three above examples.

2.2 *Wordsmyth English Dictionary-Thesaurus*

As we have seen, disagreements exist about the way a word should be segmented into syllables. A second (independent) lexical source was therefore used, namely the *Wordsmyth English Dictionary-Thesaurus*, so that our results would not be overly specialized to one particular source of data. This dictionary is also available via the World Wide Web from www.wordsmyth.net (last accessed 9 July 2004). It originated in the early 1980's when Robert Parks, a Fulbright Fellowship researcher in Japan, began to develop an English dictionary for students to use on their computers. In 1991 and 1992, the dictionary was licensed to IBM to integrate into their products, and IBM in turn supported the development of the associated thesaurus. In 1996, the University of Chicago's ARTFL (American and French Research on the Treasury of the French Language) Project assisted in presenting the first World Wide Web edition. The dictionary is composed of about 50,000 headwords covering all areas of knowledge without technical vocabulary. It provides the syllables, pronunciation, part of speech, inflected forms, and definition for each word.

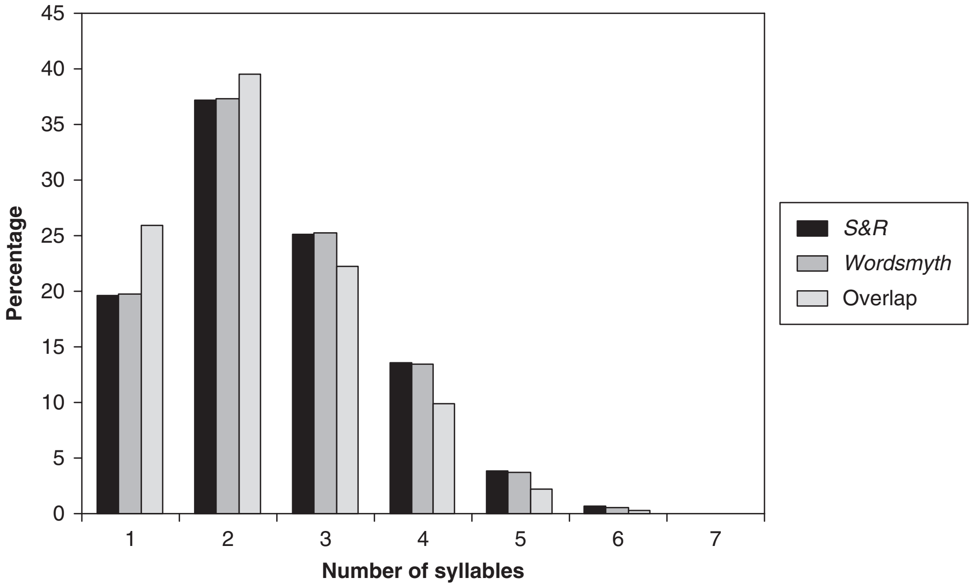
2.3 The three lexical databases

Of the 19,596 entries remaining in *Webster's Pocket Dictionary* after pre-processing to remove homonyms, 18,016 were also found in the *Wordsmyth English Dictionary-Thesaurus*, although not necessarily having the same syllabification. Since 4422 words had different syllabifications (roughly 25% of the words), this suggests that different syllabification conventions are used in compiling the two dictionaries. It is likely that the main difference is in the extent to which morphological information has been utilized in deciding syllable boundaries, although we have no easy way to confirm this.

We refer hereafter to the 18,016 words found in both dictionaries with syllabification taken from *Webster's Pocket Dictionary* as *S&R* (for "Sejnowski and Rosenberg"). This forms the first of the three lexical databases used here. The second of the three consists of these 18,016 words common to the two public-domain dictionaries with

Figure 1

Distribution of the number of syllables per word



syllabification taken from the *Wordsmyth English Dictionary-Thesaurus* (hereafter *Wordsmyth*). A third database of syllabified words (hereafter *Overlap*) was derived consisting of the 13,594 words present in both public-domain dictionaries with identical syllabification patterns in the two independent lexical sources.

2.4
Distribution of syllables in the three databases

Figure 1 shows the distribution of the number of syllables per word in the three databases of syllabified words. Most words (more than 60% of the total) have two or three syllables. There are 111,739 junctures (possible placements of syllable boundaries) in *S&R* and *Wordsmyth*, and 78,768 junctures in the *Overlap* lexicon. Of these possibilities, 26,459 (23.68% of the junctures), 26,283 (23.52% of the junctures), and

Table 2

Letter types (consonant/vowel) surrounding syllable boundaries

Structure	*S&R*	*Wordsmyth*	*Overlap*
Consonant|Consonant (%)	37.65	38.56	43.33
Vowel|Consonant (%)	31.54	36.68	33.78
Consonant|Vowel (%)	23.95	19.30	17.48
Vowel|Vowel (%)	6.86	5.46	5.41

16,808 (21.34% of the junctures) are actual syllable boundaries in *S&R*, *Wordsmyth*, and *Overlap*, respectively. Table 2 shows the letter type (vowel[4] or consonant) of the bigrams that encompass the syllable boundaries; Tables A4 and A5 in the Appendix more specifically indicate, in rank order, the 10-letter bigrams and 10-phoneme bigrams that most often surround them.

3 Description of syllabification algorithms

In this section, we briefly describe the five automatic syllabification techniques that are compared later with respect to their performance.

3.1 Rule-based systems

We have implemented and evaluated two different rule-based algorithms, namely Hammond's constraint-based parser and Fisher's implementation of Kahn's procedure. Hammond's method works in the spelling domain whereas Kahn's syllabification works in the pronunciation domain. These will now be described in turn.

3.1.1 Hammond's constraint-based syllable parser

Hammond's algorithm, based on optimality theory (OT), is guided by a constraint hierarchy that the syllabification process in the spelling domain must follow. See Hammond (1997) for theoretical background. Hammond's possible constraints are the following:

`Cpeak`	A consonant cannot be a syllable peak.
`Vmargin`	A vowel cannot be an onset or a coda.
`Onset`	Syllables must have onsets.
`Complex`	Syllable margins cannot contain more than one segment.
`Nocoda`	Syllables cannot have codas.

The first constraint entered is the highest rank and so forth. The program first breaks the given word into a list of its characters and gives each character the option of either being the onset, nucleus, or coda of a syllable in the word. It then places constraints on the word in the order given and proceeds to determine what role each character plays by following the constraints (beginning with the highest ordered constraint and ending with the lowest ordered). According to OT, words are permitted to violate constraints, but a single violation of a higher-ranked constraint is worse than any number of violations of a lower-ranked constraint.

Hammond implemented two versions of his constraint-based syllable parser in Prolog: one in 1997 and the other in 1998. These will be referred to as Hammond 1

[4] Defined by <a>, <e>, <i>, <o>, and <u>.

and Hammond 2. Both can be downloaded from the World Wide Web at http://www.u.arizona.edu/~hammond/ (last accessed 16 March 2005) but only the 1997 version offers an online user interface. These programs are based on the same theory and accept the same constraints but they do not produce identical output. The difference lies in the placement of a syllable boundary within a series of three consonants beginning and ending with vowels (e.g., in <bubble> and <poncho>). The 1997 version places the syllable boundary after the second consonant (<bubb|le> and <ponc|ho> while the 1998 version places the syllable boundary after the first consonant (<bub|ble> and <pon|cho>). The three English databases were syllabified with both versions. Hammond's original implementations could only handle a single word entered into the program, so the code was modified to process all the words of a file consecutively.

Because the five constraints can be applied in any order, there are 5! = 120 possible sets of input for Hammond's algorithm, each of which could produce different results. We ran the system with the default constraint order given by Hammond on the online interface, namely: Cpeak >> Vmargin >> Onset >> Complex >> Nocoda.

3.1.2
Fisher's implementation of Kahn's procedure

In his PhD dissertation, Kahn (1976) proposed a theory of syllabification based on a different type of constraint. Unusually, this works in the pronunciation domain. Kahn postulated that syllabification in English is derived from three categories of consonant clusters: possible syllable-initial, possible syllable-final and "universally-bad" syllable-initial (in his terminology). These consonant clusters are derived from the beginnings and endings of existing English words. For example, the two-phoneme sound /br/ is a possible syllable-initial consonant cluster because it forms the beginning of the word pronunciation /brɛd/ (<bread>) and it is therefore possible to syllabify the pronunciation /ənbreɪd/ (<unbraid>) as /ən|breɪd/. By contrast, /rk/ is considered a "universally-bad" syllable-initial consonant cluster because no English word begins with this sound combination. Therefore the pronunciation /markət/ (<market>) would have to be syllabified as /mar|kət/ and not /ma|rkət/.

Kahn's theory also permits ambisyllabicity of consonants and differences in syllable boundaries for different rates of speech (artificially slow, normal, or fast). Ambisyllabicity allows a single consonant to occupy both the last position in the coda of one syllable and the first position in the onset of the following syllable. According to this theory, the /t/ sound in /kwɔrtɜ/ (<quarter>) is ambisyllabic in normal and fast speech. Under these circumstances, / kwɔrtɜ/ would be syllabified as / kwɔrt|tɜ/ with /t/ occupying two positions.

A C implementation of Kahn's theory was developed in 1996 by William Fisher and can be downloaded from: ftp://jaguar.ncsl.nist.gov/pub/tsylb2–1.1.tar.Z (last accessed 16 March 2005). Fisher's program is accompanied by files containing lists of acceptable syllable-initial, acceptable syllable-final, and unacceptable syllable-final consonant clusters.[5] Two options are available for these lists: one has only "native" clusters, while the other includes "foreign" clusters. For example, /k/ is not

[5] In Fisher's software package, there is no explicit reference to the English lexical source that was used to define these clusters except a brief mention of June Shoup's dictionary in the early 1960s.

permissible in the native syllable-initial clusters but is permissible as a foreign cluster because it forms the beginning of /kʃatria/ (<kshatriya> is a Hindu of upper caste). Fisher further explains Kahn's "rates of speech" (in the algorithm.doc file with the download) by stating that " 'rate' is probably a conventional misnomer for degree of casualness, informality, or lack of monitoring." In this implementation, Fisher defines five different rates of speech:

(1) slow, over-precise, "syllable-by-syllable";
(2) formal, monitored, self-conscious speech;
(3) ordinary conversational speech;
(4) faster speech;
(5) fastest, sloppiest, least monitored speech.

Finally, note that we used the "tight" constraints developed during the summer of 1996 by several researchers at a Johns Hopkins CLSU speech recognition workshop (see algorithm.doc in the download).

As with Hammond's system, we changed Fisher's original implementation to process all words in a file consecutively. Because we were interested in the standard syllabification, we used the results given for the first rate (i.e., "slow, over-precise," hereafter Basic) and the third rate ("ordinary conversational speech," hereafter OCS). The program also allowed the unsyllabified input to be provided with stress information (primary, secondary and no stress) on some specific phonemes[6] and without stress information. We processed the word list both ways, using the stress information provided in *S&R* (i.e., the digit stress—see Table 1).

As mentioned earlier, the aim of Fisher's system is to find the syllabification patterns in the pronunciation domain. Therefore, it was necessary to translate the phoneme set used in his program to the phoneme set of *S&R* (a not entirely straightforward process we called "harmonization" in Damper et al., 1999). Table A3 in the Appendix gives the correspondence between these phoneme sets. Table A4 shows the five special phonemes from *S&R* that were decomposed into a two-phoneme combination to be consistent with Fisher's phoneme set. All instances of the null phoneme were removed because this special "phoneme" was not part of Fisher's set. Table A5 illustrates this recoding with and without stress information. Overall, 1338 pronunciations were recoded in *S&R* and *Wordsmyth* and 932 pronunciations were recoded in *Overlap.*

Finally, there are two modes in which Fisher's procedure can be run: with only the "native" consonant cluster constraints and with both the "native" and "foreign" consonant cluster constraints. Because the results were not improved by including the "foreign" option, we have not presented them in this article.

3.2 Data-driven methods

We now describe the three different data-driven approaches to automatic syllabification compared in this article. They are all instances of lazy learning. One of the outstanding

[6] … designated as syllabic by Fisher. These are: "ux," "ih," "ix," "ey," "eh," "ae," "aa," "aax," "s," "ao," "ow," "uh," "uw," "ay," "oy," "aw," "er," "axr," "ax," "ah," "el," "em," and "en" using his phoneme notation.

advantages of lazy learning approaches to language processing is the ease with which algorithms can be transferred to new tasks. In fact, all three methods studied here were originally designed for automatic pronunciation but are readily modified to perform syllabification. Another advantage is that, unlike eager learning, which compresses the training dataset, it is often practical to test performance on the complete dataset. As we shall see, this is more true of SbA than the other two methods.

3.2.1 Syllabification by analogy

Syllabification by analogy syllabifies words in either the orthographic or pronunciation domain. It closely follows the principles of pronunciation by analogy (PbA) set out in detail in our earlier publications (Damper & Eastmond, 1997; Marchand & Damper, 2000; Damper & Marchand, 2006, 2007).

Let us first briefly describe PbA. When an unknown word is presented as an input to the system, so-called full pattern matching between the input letter string and database entries is performed, starting with the initial letter of the input string aligned with the end letter of the database entry. If common letters are found in matching positions in the two strings, their corresponding phonemes (according to the prior alignment) and information about their positions in the input string are used to build a pronunciation lattice, as detailed next. One of the two strings is then shifted relative to the other by one letter and the matching process continues, until the end letter of the input string aligns with the initial letter of the database entry.

The pronunciation lattice is a directed graph that defines possible pronunciations for the input string, built from the matching substring information. A lattice node represents a matched letter, L^i, at some position, i, in the input. The node is labeled with its position i and the phoneme corresponding to L^i in the matched substring, P_{im}, say, for the mth matched substring. An arc is labeled with the phonemes intermediate between P_{im} and P_{jm} ($j > i$) in the phoneme part of the matched substring and the frequency count, increasing by one each time the substring with these phonemes is matched during the search through the lexicon. Arcs are directed from i to j. If the arcs correspond to bigrams, the arcs are labeled only with the frequency. (The string of phonemes intermediate between P_{im} and P_{jm} is empty.) Phonemes P_{im} and P_{jm} label the nodes at each end of the arc, that is, i and j respectively. Additionally, there is a *Start* node at position 0 and an *End* node at position equal to the length of the input string plus one.

Finally, the decision function identifies the "best" candidate pronunciation of the input according to some criterion. Possible pronunciations correspond to the string assembled by concatenating the phoneme labels on the nodes or arcs in the order that they are traversed in moving through the lattice from *Start* to *End*. If there is just one candidate corresponding to a unique shortest path, this is selected as the output. If there are tied shortest paths, five different scoring strategies are applied and the winning candidate selected on the basis of their rank (Marchand & Damper, 2000; Damper & Marchand, 2006).

The major modification in converting PbA to SbA (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 database 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 database 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 (labeling nodes), "*" and "|" are explicitly represented as possible output symbols (labeling arcs), and there is no pronunciation information labeling the nodes and arcs. From here, the process proceeds exactly as for PbA, 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 modifications to perform SbA in the pronunciation domain should now be obvious.

In our previous syllabification work using analogy (Marchand & Damper, 2007), we obtained best results by combining only three of the five scoring strategies when choosing between tied shortest paths. These were the product of arc frequencies, the frequency of the same pronunciation, and the "weak link" (see Marchand & Damper, 2000, and Damper & Marchand, 2006, for full specification). Accordingly, in this work, these same three scoring strategies are used exclusively, and combined by rank fusion[7], for SbA.

Although the required program code 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 syllabified database at l–1 junctures. This leads to a significant increase in run times for SbA relative to PbA.

3.2.2 Look-up procedure

This method was originally proposed by Weijters (1991) as a means of letter-to-phoneme conversion (i.e., automatic pronunciation), where it was shown to be superior to NETtalk, the well-known neural network of Sejnowski and Rosenberg (1987). It was then adapted for the syllabification process and presented in the comparison of syllabification algorithms for Dutch spellings by Daelemans and van den Bosch (1992). This method can syllabify words in either the spelling domain or the pronunciation domain. The first step is to construct a table encoding the knowledge implicit in the training set by converting each syllabified entry into a series of N-grams. Each N-gram has a left and right context and a central "focus" character. The length of the N-gram (i.e., N) is equal to the sum of the sizes of the left and right contexts plus one (the focus character).

7 … using Borda counting, see Damper and Marchand (2006).

For example, if the syllabified word <kid|ney> is part of the (spelling domain) training corpus, then with a left context of 1 character and a right context of 2 characters, the *N*-grams (or 4-grams in this case) for this word would be: <–kid>, <kidn>, <idne>, <dney>, <ney–> and <ey– –>. That is, to allow every character to be a focus character, there is an *N*-gram for each character in a word. When the focus character has no left context (as in <–kid >) or right context (as in < ey– –>), the character positions in the context are filled with nulls. Each *N*-gram is stored in the table along with the corresponding juncture class, that is, the syllabification information. This consists of a field stating whether or not a syllable boundary exists immediately after the focus character in the *N*-gram (SB or NSB for syllable boundary and non-syllable boundary, respectively). Table 3 shows how the *N*-grams in the word <kid|ney> are stored in the look-up table.

Table 3

Example of look-up table construction with the word <kid|ney>

Left context	*Focus character*	*Right context*	*Juncture class*
–	k	id	NSB
k	i	dn	NSB
i	d	ne	SB
d	n	ey	NSB
n	e	y–	NSB
e	y	– –	NSB

Once the construction of the look-up table is complete, words for which the syllabification is unknown can be syllabified based on the information in the table. Input words are broken down into a set of *N*-grams in the same manner as described above for table construction. The table is then searched for the closest matches to each *N*-gram. When found, closest matches are examined to determine whether the majority has, or does not have, a syllable boundary following the focus character. If the majority has a syllable boundary, a syllable boundary is placed at the appropriate position in the word; otherwise, a non-syllable boundary is placed at that position.

For example, if the word <midnight> is to be syllabified by this method, one of the *N*-grams to match would be <midn>. The closest match in the table is <kidn>, which has a syllable boundary following the focus character <d>. A syllable boundary is then placed following the focus <d> in <midn> and, therefore, <midnight> is determined to be syllabified as <mid|night>

The process of determining which *N*-grams in the pre-compiled look-up table best fit a given *N*-gram is described in Algorithm 1. Here, `NgramT` is a given *N*-gram stored in the table and `NgramS` is an *N*-gram to be syllabified. It follows that `NgramT[i]` is the *i*th position in the *N*-gram (for example, `NgramT[1] = m` when `NgramT` is <midn>). The closest-fit *N*-grams are those with the highest `MatchValue`.

Algorithm 1: Computation of best-fit *N*-grams in table look-up method

```
FindMatchValue(weights, NgramT, NgramS)
MatchValue:= 0
for i:= 1 to length(weights) do
   if (NgramT[i] = NgramS[i]) then
       MatchValue:= MatchValue + weights[i]
   end if
end for
```

We ran the look-up procedure using all 15 different sets of weights presented in the original description of the method (Weijters, 1991, p.1647). These weight vectors are given in Table A6 in the Appendix.

3.2.3 Exemplar-based generalization

The version tested here (also known as IB1-IG) is due to Daelemans, van den Bosch and Weijters (1997). It operates in a manner similar to the look-up procedure with the only difference being the weights used to determine the closest-fit *N*-grams. In this method, the weights are calculated with a function that determines the relative importance of each position in the *N*-gram (i.e., letter or phoneme positions in the spelling and pronunciation domains, respectively). The process of determining the weights is based on the concept of information entropy by using information from the table of stored *N*-grams. Each position in an *N*-gram is considered to contribute a real-valued amount of information to the process of determining the placement of a syllable boundary. This value can be determined via the series of steps presented below.

First, the entropy of the entire table of *N*-grams extracted from the training corpus is calculated. Essentially, Daelemans, van den Bosch, and Weijters define database (or look-up table) information entropy as the number of bits of information needed to know the decision whether a syllable boundary should be placed after the focus character or not given a pattern or (*N*-gram). This is calculated as:

$$E(D) = \sum_{i=1}^{2} P_i \log_2 P_i \tag{1}$$

where $E(D)$ is the information entropy of database D, P_1 is the probability of an *N*-gram being associated with a syllable-boundary decision, and P_2 is the probability of an *N*-gram being associated with a non-syllable-boundary decision. As there are only two possibilities—to place or not to place a syllable boundary after the focus character—equation (1) can also be written as:

$$E(D) = -\left(\frac{N_S}{N_T}\right)\log_2\left(\frac{N_S}{N_T}\right) - \left(\frac{N_{\neg S}}{N_T}\right)\log_2\left(\frac{N_{\neg S}}{N_T}\right) \tag{2}$$

where N_S is the number of stored *N*-grams that have a syllable boundary following the focus character, N_S is the number of stored *N*-grams that do not have a syllable boundary following the focus character, and N_T is the number of stored N-grams (i.e., $N_S + N_S$).

From equation (2), the information gain of each position in an N-gram can now be determined. This requires two additional equations. The first computes the average information entropy at position f in an N-gram, $E(D_f)$, by taking the information entropy of the database (or table) restricted to each possible value (or character) for the position in the N-gram. This is given by:

$$E(D_f) = \sum_{c \in V} E(D_f) \frac{\text{card}(D_{f=c})}{\text{card}(D)} \tag{3}$$

where $D_{f=c}$ is the set of those N-grams in the table that have character c at position f, V is the set of characters that occur at position f in an N-gram, and card( ) is the cardinality of a set (i.e., card(D) is the total number of N-grams in database D).

The second equation necessary for calculating the information gain $G(f)$ at a given position f in an N-gram is:

$$G(f) = E(D) - E(D_f) \tag{4}$$

To run this method, we first followed Daelemans, van den Bosch, and Weijters and used the same values of N as in their work, namely 3, 5 and 7 with the focus letter in the middle of the N-gram. In addition to these values, we extended the study to use N-grams of size 9 and 11 (with left and right contexts of 4 and 5, respectively).

3.3 Speed of the algorithms

Because the various algorithms have been run on several different machines, we do not have precise timings, but some useful comments on relative efficiency can be made. Generally, the methods satisfy the following relationship, which include both training (where required) and test times:

$$\text{times(rules)} < \text{time (look-up table)} < \text{time (SbA)} < \text{time(IB1-IG)} \tag{5}$$

Neither the rules nor SbA require prior training. The rules are very fast to apply, but SbA requires significant computation to build and traverse the syllabification lattice (see Marchand & Damper, 2007 for additional comments on the computational complexity of SbA). The time for the look-up table and IB1-IG methods depends on the window size. Obviously, the larger is the window size, the longer the algorithm takes to run. A good deal of the apparent inefficiency of the table look-up and IB1-IG methods stems from the need for repeated re-training when using leave-one-out testing (see next section), which would not be required in a practical setting.

4 Results

Results were computed using word and juncture accuracy. Word accuracy is simply the number of words syllabified by the method in exactly the same way as is given by the lexicon used. Juncture accuracy compares syllabification at the sub-word level. Each position between letters is assessed to determine whether it was classified correctly. For example, the word <satire> has five junctures: <s*a*t*i*r*e>. The syllabification according to *S&R* is <sa|tire>. If an algorithm syllabifies the word as <sat|ire>, this is considered entirely wrong in terms of word accuracy. However, it is 60% correct in terms of juncture accuracy, as three out of five junctures are correct.

For the rule-based methods, there is no difficulty in evaluating syllabification performance on each of the three datasets in their entirety. For data-driven methods, however, this is not always possible. Eager learning approaches such as error back-propagation training of neural nets, for example, require the available data to be split into disjoint training and test sets (and possibly a validation set also). However, the lazy learning methods employed in this study all allow performance to be evaluated on the complete dataset without undue difficulty. For SbA, we used the well-established leave-one-out procedure, whereby each word is removed from the corpus in turn, and its syllabification inferred from the remaining words. For the table look-up and exemplar-based methods, we again remove each word in turn from the corpus, but this time it is necessary to build the respective tables repeatedly before testing each word. Although this is computationally expensive, it is not prohibitively so (table construction being generally faster than error back-propagation training, for instance.)

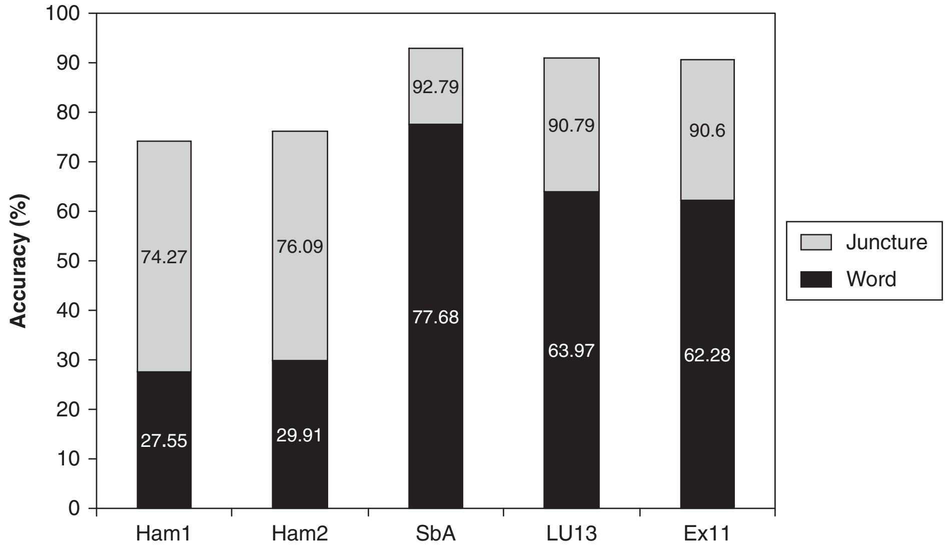
Figures 2, 3 and 4 show the results in both spelling and pronunciation domains for the various automatic syllabification methods on the *S&R*, *Wordsmyth*, and *Overlap* databases, respectively. For Hammond 1 and 2, results could only be obtained in the spelling domain. For table look-up, the set of weights that provided broadly the best results (for each database) are presented in the figures. Results were obtained for N-grams from $N = 5$ up to $N = 11$ for the exemplar-based approach. As expected, results were poor for $N = 3$ as insufficient context is captured around the focus letter, and by $N = 11$ the algorithm was very slow and there was some indication that performance was falling off. As with table look-up, the results for broadly the best value of N are presented in the figures. For the Fisher/Kahn system, results can only be obtained in the pronunciation domain. There was no difference between the results when stress was provided and when it was not for the Basic (slow) rate of speech. However, this was not the case for the ordinary conversational speech condition, where the inclusion of stress gave a large improvement. There were no ambisyllabic results in our data (although the implementation by Fisher is capable of giving them as output).

The pattern of results is consistent across dictionaries. Generally, performance in the pronunciation domain is well above that in the spelling domain. The rule-based methods (Hammond and Fisher/Kahn) are vastly poorer than the data-driven methods. For instance, the performance for Hammond 2 is approximately 30–35% words correct in the spelling domain. One factor strongly influencing this is the algorithm's very poor showing on identifying boundaries between vowel letters (just 20–30%). We do not think such rule-based methods are at all usable as a complete solution in computational linguistics and/or speech technology, although hybrid rule-based/data-driven systems may possibly be useful.

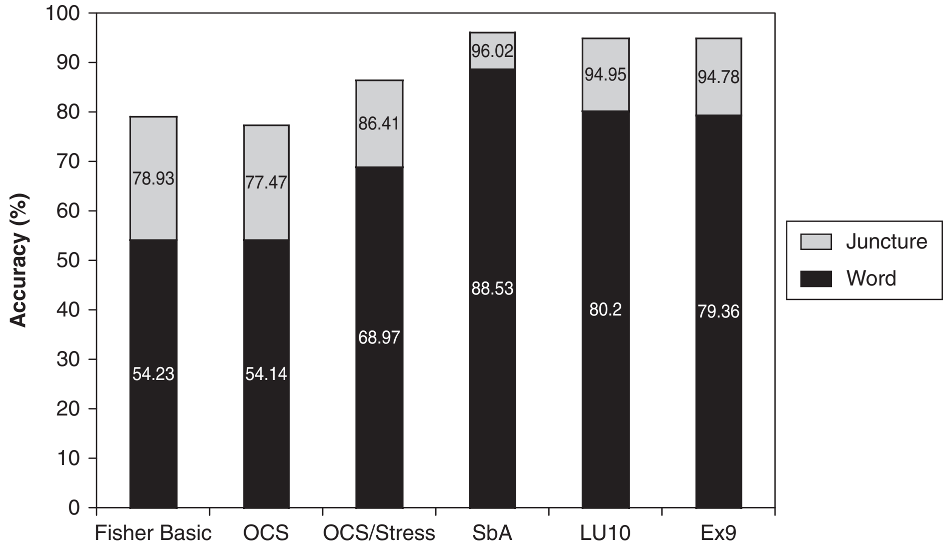
Of the data-driven methods, it is difficult to choose between the best table look-up and exemplar-based results: the latter does better on two of the three dictionaries

Figure 2

Syllabification results (percentage correct) on *S&R* in (a) spelling and (b) pronunciation domains. KEY: Ham1, Hammond 1; Ham2, Hammond 2; SbA, syllabification by analogy; LU10, look-up table, version 10; LU13, look-up table, version 13; Ex9, exemplar-based, $N = 9$; Ex11, exemplar-based, $N = 11$



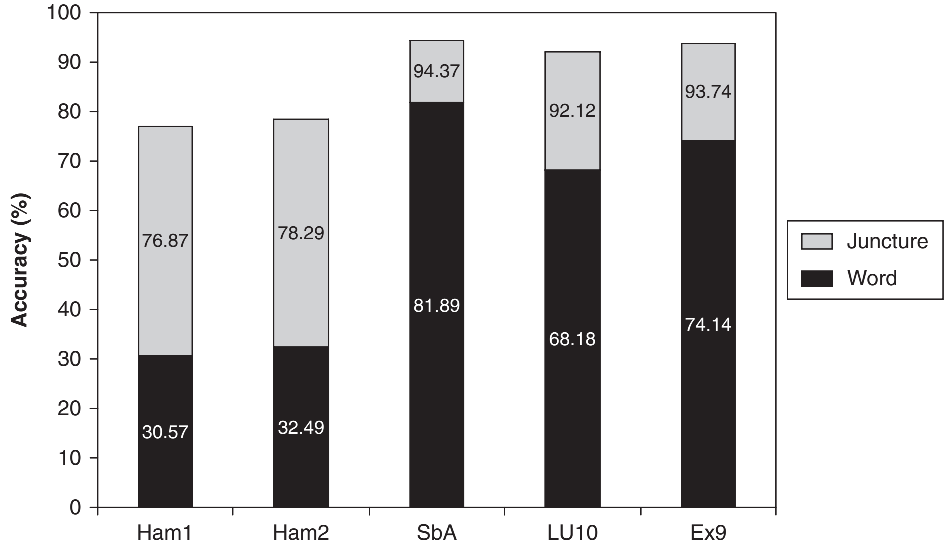
(a) Spelling domain



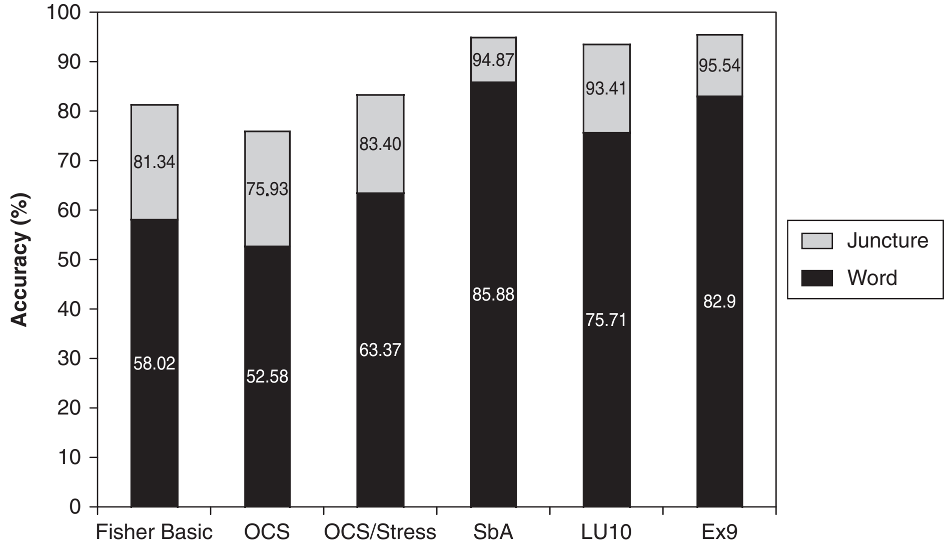
(b) Pronunciation domain

Figure 3

Syllabification results (percentage correct) on *Wordsmyth* in (a) spelling and (b) pronunciation domains. Key: Ham1, Hammond 1; Ham2, Hammond 2; SbA, syllabification by analogy; LU10, look-up table, version 10; Ex9, exemplar-based, $N = 9$



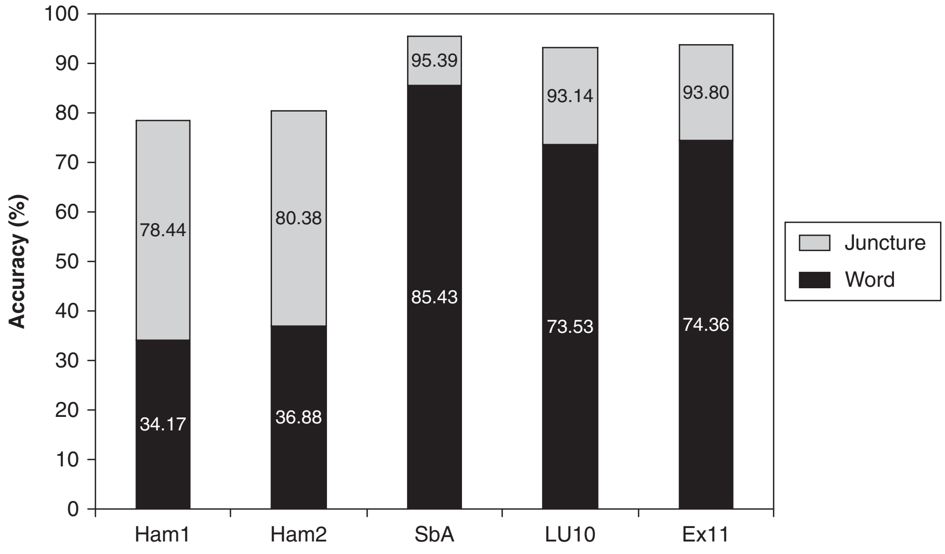
(a) Spelling domain



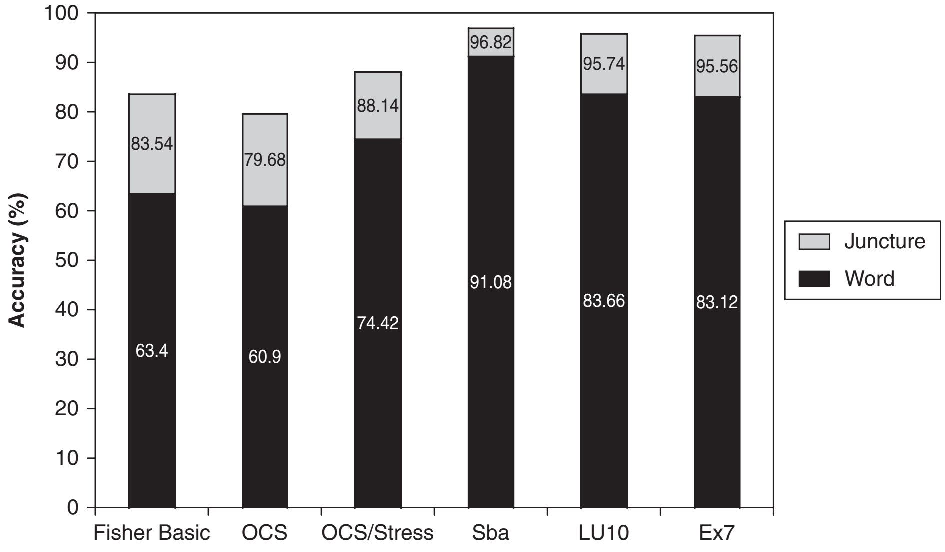
(b) Pronunciation domain

Figure 4

Syllabification results (percentage correct) on overlap in (a) spelling and (b) pronunciation domains. Key: Ham1, Hammond 1; Ham2, Hammond 2; SbA, syllabification by analogy; LU10, look-up table, version 10; Ex7, exemplar-based, $N = 7$; Ex11, exemplar-based, $N = 11$



(a) Spelling domain



(b) Pronunciation domain

in the spelling domain, but this is reversed in the pronunciation domain. The most striking result, however, is the obvious superiority of SbA.

Although the statistical significances of the observed performance values has not been assessed, the very large number of degrees of freedom involved (there are thousands of words in each corpus) means that even apparently quite small differences in percentages of words correct can be enormously significant. See Marchand and Damper (2007) for more detailed argument on this point.

For all methods and both spelling and pronunciation domains, non-syllable boundary identification is less error prone than syllable boundary detection (results not shown in detail). For example, for SbA and the *Overlap* lexicon in the spelling domain, correct syllable boundary placement rate was 87.53%, whereas for non-syllable boundary placement, it was 97.52%. It seems that all methods are conservative in their placement of syllable boundaries, which are rarer than non-syllable boundaries, resulting in a preponderance of false negative errors over false positives.

We also looked at the pattern of errors that each procedure made. In the spelling domain, the most common cause of errors was incorrect identification of the boundary between letters <a> and <t>. This was also the most common bigram having a syllable boundary, see Table A1. Taking table look-up, version 13, and the *S&R* database as an illustrative example, there are 1270 instances of <a*t> of which 93.46% were correctly identified as non-syllable boundaries (cf. 95.16% across all non-syllable boundaries). There were 632 instances of <at> of which 81.65% were correctly identified as syllable boundaries (cf. 76.70% across all syllable boundaries). As an example of errors made, *S&R* has the entries <cat|al|yst> and <sa|tire> but the look-up table method gave <ca|tal|yst> and <sat|ire>, respectively.

In the pronunciation domain, the most common cause of errors was incorrect identification of the boundary between phonemes /x/ (the schwa vowel in *S&R* notation) and /s/. As can be seen in Table A2, the schwa vowel is very common adjacent to a syllable boundary, although /x|s/ is not itself especially common. For /x*s/, there were 853 instances of which 95.19% were correctly recognized as non-syllable boundaries (cf. 96.70% in the general case). For /x|s/, there were 331 instances of which just 83.69% were correctly recognized (cf. 90.51% in general). As an example of errors made, *S&R* has the entries /x|sAd/ (<aside>) and /kxs|Et/ (<cassette>) but the look-up table method gave /xs|Ad/ and /kx|sEt/, respectively.

In a speech technology application, specifically in synthesis, a syllabification error in the orthography is likely to result in a wrong pronunciation of the word; whereas a wrong syllable boundary inserted in the (already given) phone symbol string may result in wrong segmental durations and/or inappropriate aspiration. It is unknown which of these would be the more harmful for intelligibility and/or quality of the synthetic speech.

5 Discussion and conclusions

Automatic syllabification is an important but difficult problem bearing on issues such as pronunciation generation for text-to-speech synthesis and pronunciation modeling in speech recognition. As yet, there is no accepted method for automatic syllabification in the literature. Part of the difficulty stems from the lack of a sound definition of the syllable in linguistic theory, making it hard to design and evaluate algorithms for this purpose. Objective evaluation requires that we have some reasonably-sized corpus of already-syllabified words that can act as a "gold standard" against which to assess correctness of the algorithm's output. At present, no widely-accepted gold standard exists.

There are essentially two possible approaches to automatic syllabification: rule-based and data-driven. In this work, we have compared two rule sets based on expert knowledge and three data-driven methods based on automatic inference from a corpus of already-syllabified words. Hence, in the latter case, the issue of a gold standard arises once more. To address this important issue, we attempt to use multiple, independent dictionaries of syllabified words and then to look for consistency or consensus among results as evidence that the syllabifications are broadly correct. We also use the "overlap" or conjunction of entries in the different dictionaries as a separate corpus that ought to be closer to a gold standard than any of the individual contributions, since it will not include words on which the latter disagree (which is likely to point to idiosyncrasies among dictionaries). In this work, we have used two independent dictionaries (*S&R* and *Wordsmyth*) and their overlap. The five methods studied are the rule sets from Hammond and Fisher/Kahn, a table look-up method due to Weijters, the exemplar-based method of Daelemans, van den Bosch and Weijters and syllabification by analogy (SbA) from Marchand and Damper. In each case, performance is evaluated across the whole of each available corpus, something that is easier to achieve with rule-based and lazy learning techniques (as used here) than with eager learning techniques such as decision trees and neural networks (which require held-out testing data). This is done in both spelling and pronunciation domains, although it should be noted that the two rule-based methods are specialized to a particular domain (and cannot be tested on both).

The knowledge-based rule sets perform poorly compared to the data-driven methods. We do not think such methods are competitive to the extent that there is no case for using them in practical applications (unless possibly in conjunction with data-driven approaches). Among the data-driven methods, SbA is easily the best. As far as the different dictionaries are concerned, best performance is obtained on the *Overlap* database. There are probably two reasons for this. First, the *Overlap* dictionary excludes many polysyllabic words (see Figure 1), because there is less consensus in the syllabification of these words. These are harder to syllabify correctly. Second, the overlap process removes idiosyncratic entries from *S&R* and *Wordsmyth*.

We believe there are sound reasons to expect the pattern of results seen here. Our earlier study of the strongly-related problem of automatic pronunciation (Damper et al., 1999) showed exactly the same trends. Data-driven methods outperformed rule-based techniques by an enormous margin, lazy learning was superior to eager learning, and the analogy method outperformed the other lazy learning approach studied (the exemplar-based IB-IG1 technique). In our opinion, expert rule-based approaches suffer many drawbacks, including lack of conformance with real data, the limited ability of human experts to distinguish real from apparent regularities in very large datasets (like the effectively unbounded whole of natural language), and a tendency to over-rate dramatically the strength of weak, tentative linguistic theories. There are also good theoretical and empirical reasons to expect analogy to outperform other methods in language processing tasks. As Pirrelli and Federici (1995, p.855) write: “the space of analogy is … eventually more accurate than the space of rules, as the former, but not the latter, is defined by the space of already attested base objects.” In other words, inference is based on real and specific examples rather than on imagined, general cases. Unlike the other two data-driven methods studied here, analogy does not use a fixed-size window on the input text, but varies the context according to the partial matches found with the “already attested base objects.” Input/output mappings are modeled together in variable-size chunks, so handling long-range dependencies easily and naturally. In a real sense, analogy (at least, as implemented in this work) is the laziest of lazy techniques, retaining the evidence base in full, with no compression whatsoever. This means that “exceptional” items are kept entire and can be profitably used in inferencing.

An example from automatic pronunciation should help to make the point. Consider the exceptional word of English <quay>. Employing an eager learning technique like error back-propagation training of NETtalk, which encodes spelling-sound regularities in terms of network connection weights, we would expect the pronunciation of this atypical word to be *over*-regularized. Hence, if we then encounter the unknown word <quayside> in the input, it will be mispronounced. With analogy, however, the words <quay> and <side> remain uncompressed in the evidence base, so it is in principle possible that <quayside> will be pronounced correctly. In the words of Daelemans, van den Bosch, and Zavrel (1999, p.42): “keeping full memory of all training instances is at all times a good idea in language learning.” This is precisely what analogy does, helping in part to account for its superiority over the other methods studied in this article.

References

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

AHA, D. W., KIBLER, D., & ALBERT, M. (1991). Instance-based learning algorithms. *Machine Learning*, **6**(1), 37–66.

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*. London: André Deutsch.

DAELEMANS, W., & van den BOSCH, A. (1992). Generalisation performance of backpropagation learning on a syllabification task. In M. F. J. Drossaers & A. Nijholt (Eds.), *TWLT3: Connectionism and natural language processing* (pp.27–37). Enschede, The Netherlands: Twente University.

DAELEMANS, W., van den BOSCH, A., & WEIJTERS, T. (1997). IGTree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, **11**(1–5), 407–423.

DAELEMANS, W., van den BOSCH, A., & ZAVREL, J. (1999). Forgetting exceptions is harmful in language learning. *Machine Learning*, **34**(1–3), 11–43.

DAMPER, R. I., & 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. (2006). Information fusion approaches to the automatic pronunciation of print by analogy. *Information Fusion*, **71**(2), 207–220.

DAMPER, R. I., MARCHAND, Y., ADAMSON, M. J., & 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., & 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.

GOSLIN, J., & FLOCCIA, C. (2007). Comparing French syllabification in preliterate children and adults. *Applied Psycholinguistics*, **28**(2), 341–367.

GOSLIN, J., & FRAUENFELDER, U. H. (2001). A comparison of theoretical and human syllabification. *Language and Speech*, **44**(4), 409–436.

HAMMOND, M. (1997). Optimality theory and prosody. In D. Archangeli & T. Langendoen (Eds.), *Optimality theory: An overview* (pp.33–58). Cambridge, MA: Blackwell.

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., & 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., & MÖBIUS, B. (1998). Multilingual syllabification using weighted finite-state transducers. In *Proceedings of 3rd European Speech Communication Association* (*ESCA*)/ *COCOSDA international workshop on speech synthesis* (pp.71–76). Jenolan Caves, Australia. European Speech Communication Association.

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

LIANG, F. M.1983. *Word hy-phen-a-tion by com-put-er*. Palo Alto, CA: Department of Computer Science, Stanford University.

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

MARCHAND, Y., & DAMPER, R. I. (2007). Can syllabification improve pronunciation by analogy? *Natural Language Engineering*, **13**(1), 1–24.

MÜLLER, K. (2001). Automatic detection of syllable boundaries combining the advantages of treebank and bracketed corpora training. In *Proceedings of 39th annual meeting of the association for computational linguistics* (pp.402–409). Toulouse, France.

MÜLLER, K., MÖBIUS, B., & PRESCHER, D. (2000). Inducing probabilistic syllable classes using multivariate clustering. In *Proceedings of 38th annual meeting of the association for computational linguistics* (pp.225–232). Morristown, NJ: Association for Computational Linguistics.

PIRRELLI, V., & FEDERICI, S. (1995). You'd better say nothing than something wrong: Analogy, accuracy and text-to-speech applications. In J. Pardo (Ed.), *Proceedings of 4th European conference on speech communication and technology, Eurospeech'95* (Vol. 1, pp.855–858). Maarid, Spain. European Speech Communication Association.

PULGRAM, E. (1970). *Syllable, word, nexus, cursus*. The Hague: Mouton.

RUMELHART, D. E., HINTON, G. E., & WILLIAMS, R. (1986). Learning representations by back-propagating errors. *Nature*, **323**(9), 533–536.

SEJNOWSKI, T. J., & 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 & N. Smith (Eds.), *The structure of phonological representations* (Vol. 2, pp.337–383). Dordrecht: Foris.

TREIMAN, R., BOWEY, J. A., & BOURASSA, D. (2002). Segmentation of spoken words into syllables by English-speaking children as compared to adults. *Journal of Experimental Child Psychology*, **83**(3), 213–238.

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

WEIJTERS, A. (1991). A simple look-up procedure superior to NETtalk? In T. Kohonen, K. Mäkisara, O. Simula, & J. Kangas (Eds.), *Proceedings of international conference on artificial neural networks (icann-91)* (Vol. 2, pp.1645–1648). Amsterdam: Elsevier Science.

Appendix

Table A1
Letter bigrams that most often surround syllable boundaries

S&R		*Wordsmyth*		*Overlap*	
Bigram	Percentage	Bigram	Percentage	Bigram	Percentage
A|T	2.39	A|T	3.16	A|T	2.77
T|I	2.12	N|T	2.66	N|D	1.95
N|D	1.65	I|T	2.54	N|T	1.92
I|A	1.64	S|T	1.74	L|L	1.84
L|L	1.61	I|A	1.61	S|T	1.71
N|T	1.59	I|C	1.51	I|A	1.65
S|T	1.44	L|L	1.50	I|C	1.51
E|R	1.40	N|D	1.46	C|T	1.47
T|E	1.40	A|B	1.30	N|S	1.46
R|I	1.40	L|I	1.26	N|C	1.34
Total	16.64		18.74		17.62

Table A2
Phoneme bigrams that most often surround syllable boundaries

S&R		*Wordsmyth*		*Overlap*	
Bigram	Percentage	Bigram	Percentage	Bigram	Percentage
x|r	2.82	x|t	3.33	e|S	2.25
l|x	1.97	n|t	2.28	i|x	1.92
i|x	1.96	I|x	2.00	n|d	1.92
e|S	1.72	i|x	1.96	x|l	1.89
x|l	1.68	e|S	1.73	l|x	1.86
x|b	1.68	r|x	1.72	n|s	1.84
r|i	1.62	s|t	1.70	s|t	1.77
n|d	1.62	x|b	1.67	x|n	1.62
t|x	1.60	n|s	1.54	x|r	1.61
x|t	1.59	x|l	1.48	x|b	1.52
Total	18.26		19.41		18.20

Table A3
Harmonization scheme used to map *S&R* phoneme set onto Fisher's set

Phoneme in S&R	*Phoneme in Fisher's system*	*As in …*	*Phoneme in S&R*	*Phoneme in Fisher's system*	*As in …*
i	iy	beat	n	n	night
u	ux	blue	N	en	button
I	him	bit	G	nx	king
I	ix	roses	s	s	sigh
E	eh	bet	p	p	pet
@	ae	bat	b	b	bet
a	aa	bob	t	t	tot
a	aax	calm	t	dx	batter
c	ao	bought	d	d	dot
o	ow	boat	k	k	kite
U	uh	book	g	g	got
u	uw	boot	e	ey	bait
A	ay	bite	C	ch	check
O	oy	boy	!	ts	tse-tse
W	aw	bout	J	jh	jet
R	er	pert	f	f	fed
R	axr	butter	v	v	vote
x	ax	abut	T	th	thigh
^	ah	abut	D	dh	thy
*	wh	which	z	z	zoo
w	w	witch	S	sh	shed
y	y	yet	Z	zh	measure
r	r	right	h	hh	hot
l	l	light	m	m	might
L	el	bottle	M	em	bottom

Table A4
Phoneme recodings used in this study

S&R phoneme	*Example*	*Number of occurrences*		*Two phonemes in Fisher*	*Recoding in S&R*
		S&R and Wordsmyth	*Overlap*		
K	sexual	15	10	k + sh	k + S
X	excess	336	263	k + s	k + s
Y	cute	959	648	y + uh	y + U
#	examine	43	24	g + z	g + z
+	memoir	20	12	w + axr	w + R

Table A5
Two examples of recoded pronunciations. New phonemes are shown in bold

Initial coding in S&R		*New coding for Fisher's system*	
Pronunciation	Stress pattern	Without stress	With stress
I#@ mxn- <examine>	0 < 1 < 0 < <	**Igz**@mxn	0**Igz**1@m0xn
mEm+-r <memoir>	> 1 < 2 < <	mEm**WR**r	m1Em**W**2**R**r

Table A6
Weight vectors used in the look-up procedure

Version	*Left context*				*Focus character*	*Right context*				
	–4	–3	–2	–1	0	+1	+2	+3	+4	+5
1					1					
2				1	4					
3					4	1				
4				1	4	1				
5			1	4	16	4				
6				4	16	4	1			
7			1	4	16	4	1			
8			1	4	16	4	2			
9		1	4	16	64	16	4	1		
10		1	4	16	64	16	5	1		
11	1	4	16	64	256	64	17	4		
12		4	16	64	256	64	16	4	1	
13		4	16	64	256	64	17	4	1	
14		16	64	256	1024	256	64	16	4	1
15		16	64	256	1024	256	65	16	4	1