# Learning English Syllabification Rules

Jian Zhang and Howard J. Hamilton

Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada, S4S 0A2
e-mail: {*jian*, *hamilton*}@cs.uregina.ca

**Abstract.** This paper describes LE-SR (Learning English Syllabification Rules), the first machine learning program that learns English *syllabification rules*, i.e., rules that tell how to divide English words into syllables for pronunciation. LE-SR uses a unique knowledge representation called C-S-CL-SS which effectively generalizes English graphemes. Given a 20,000 on-line pronouncing dictionary, LE-SR learned 423 syllabification rules from 90% of instances that have a predictive accuracy of 90.35% on the unseen 10% instances.

## 1 Introduction

LE-SR (Learning English Syllabification Rules) is the first machine learning program to learn English *syllabification rules*, i.e., rules that tell how to divide English words into syllables for pronunciation. The goal is produce a small number of syllabification rules that have high accuracy on *unseen words* (i.e., words that were not present in the training instances used to create the rules). Such rules are of interest to linguists, could be used in a speech synthesis system, and have been identified as essential to the LEP (Learning English Pronunciation) project.

LE-SR is one of four elements in the LEP project, which is a purely symbolic approach to learning English pronunciation based on separate steps for learning grapheme segmentation, syllabification, stress for syllables, and grapheme-to-phoneme translation [Zhang and Hamilton, 1996]. The overall approach is sketched in Figure 1. Four rule sets are learned separately and then applied in sequence. Currently, each component (LE-GS, LE-SR, LE-SS, or LE-PG) learns rules for one task directly from an on-line pronouncing dictionary. Future research will study interactive learning among the four learning components.

In a previous report, we described LE-SR 1.0, which learns syllabification rules based on counting the frequencies of inter-syllabic patterns [Zhang and Hamilton, 1997]. In this paper, we present LE-SR 2.2, which uses the Iterated Version Space Algorithm (IVSA) to generate, select, and order a set of syllabification rules; IVSA is described in [Hamilton and Zhang, 1996].

To explain the LE-SR approach, we give our definitions for graphemes and phonemes. A *phoneme* is the smallest unit of any spoken language that distinguishes sounds and meanings of words. For example, the phonemes /æ/ and /Λ/

distinguish the words *cat* and *cut*. A *grapheme* is a letter or a combination of letters that represents one sound unit (phoneme, cluster, or silent morphopheme). For example, the word "friend" has 6 letters but 5 graphemes: <f>, <r>, <ie>, <n>, and <d>.

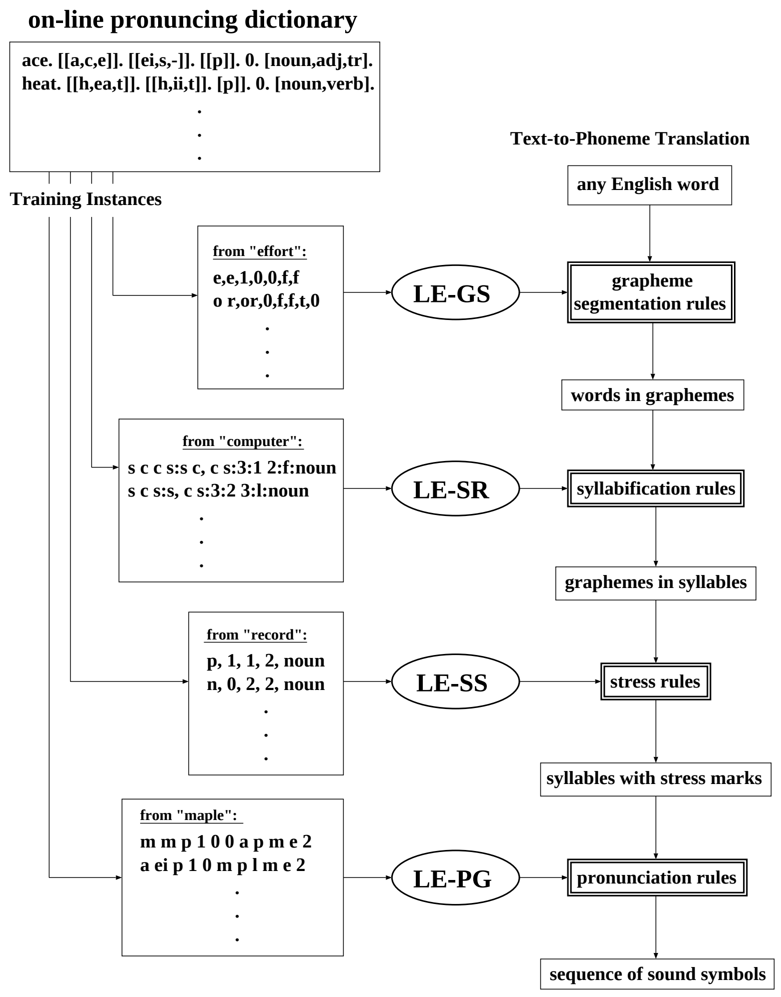


**Fig. 1.** The LEP Project

In the text-to-speech literature of the last 20 years, word syllabification has been ignored. Ling and Wang use a grapheme-to-phoneme method to translate all graphemes into phonemes [Ling and Wang, 1995]; PDtalk [Mudambi and Schimpf, 1994], DHtalk [Hochberg et al., 1991], NETtalk [Sejnowski and Rosenberg, 1987], DECtalk [Klatt, 1987], KLATtalk [Klatt, 1982], and NRLtalk [Elovitz et al., 1976] use letter-to-phoneme methods to translate each letter of a word into a phoneme; MITtalk [Allen et al., 1987] translates all morphs into

| Symbol | Grapheme Type | Class | Graphemes / Cluster |
|---|---|---|---|
| C | consonant | − | b c d f g h j k l m n p q r s t v w c x z |
|  |  |  | bb cc cch ch ck cq dd dg dj ff gg gh gm |
|  |  |  | gn kk kn ll mb mm mn ng nn ph pn pp ps |
|  |  |  | rr sc sh sl ss st tch th ts tt vv wh wr zz |
| S | syllabic | + | a i o u w c - aa ae ai ao ar au aw ay ea |
|  |  |  | ear eau ee ei eo er err eu eur ew ey ia |
|  |  |  | ie ier ieu iew io iou ir irr oa oar oe oi |
|  |  |  | oir oo or orr ou our ow oy re ua ue uer |
|  |  |  | ui uo uor ur ure urr uu uy yr |
| CL | cluster | − | bl br bbr bbl cl ckl cr dr fl fr gl gr |
|  |  |  | kr pl pr tr |
| SS | special syllabic | +/− | e y |

**Table 1.** The C-S-CL-SS Representation

phonemes. With these approaches, words in input text are not pronounced as syllables but as single sounds, and stresses are marked on individual vowels instead of syllables. This is one reason that the word utterance of commercial speech synthesizers does not sound as natural as human speech. Using the syllabification rules learned by LE-SR to divide any English word into syllables, a text-to-speech system could utter a word (regardless of whether it has been seen before) according to syllables instead of individual letters. Previous approaches to speech synthesis have either obtained syllabification from a dictionary or used a few hand-coded rules.

The remainder of this paper is organized as follows. In Section 2, syllabification is explained. Section 3 describes our approach by formulating the task of creating syllabification rules as a learning problem. A descriptive example is given in Section 4, and experimental results are presented in Section 5. Conclusions and suggestions for future research are given in Section 6.

## 2 Syllabification

For centuries, linguists of English have analyzed the syllables of English words. They discovered that an English syllable can have "zero, one, two or three consonants before the vowel and from zero to four consonants following the vowel" ([MacKay, 1987], pp. 41). This makes syllabification of English words very difficult. We have not found a piece of research that presents a complete set of English syllabification rules, although partial sets of rules are often given in linguistic books [Ladeforged, 1982], [MacKay, 1987], [Kreidler, 1989], [O'Grady and Dobrovolsky, 1992].

Informally, most English speakers know that a syllable must contain a vowel sound. Precisely, a written syllable must contain a *syllabic grapheme* that corresponds to a vowel sound. A syllabic grapheme is the "centre of a syllable" [Kreidler, 1989]. Vowel graphemes are usually syllabic (denoted "+syllabic"), but some can be non-syllabic ("–syllabic"), such as <e>, which can be silent. As well, the semi-vowel <y> is sometimes pronounced as a vowel and sometimes as a consonant. Some English speakers and dictionaries treat <m>, <n>, and <l> as syllabic consonants in some contexts, while others treat them as ordinary consonants in all contexts.

A syllable may include zero or more consonants. For example, the word *a* consists of one syllable, and this syllable contains the vowel sound /∂/. The word *syllabic* has three syllables, containing the /i/, /æ/, and /i/ sounds, which correspond to the syllabic graphemes <y>, <a>, and <i>, respectively. These examples suggest that identifying syllabic graphemes for English is essential for learning syllabification rules.

When formulating syllabification rules, we believe that is useful to extend the classification of +syllabic and −syllabic into a four-part classification, called the *C-S-CL-SS representation*. The symbols C, S, CL, and SS represent the class of phonemes that can be represented by the graphemes. In Table 1, we show the graphemes in each class, according to NETalk Corpus 2 (NTC2), which is an on-line pronouncing dictionary [Zhang et al., 1997]. Graphemes classified as C always correspond to consonant phonemes, and those classified as S always correspond to vowel phonemes in the dictionary. The symbol CL represents a *consonant cluster*, which here is treated as a sequence of consonant graphemes that frequently occur together. For convenience of expression, we refer to consonant clusters as if they were single graphemes, although they are not. SS represents a grapheme that corresponds to either a consonant or a vowel phoneme depending on context. The relationships are summarized in Figure 2.

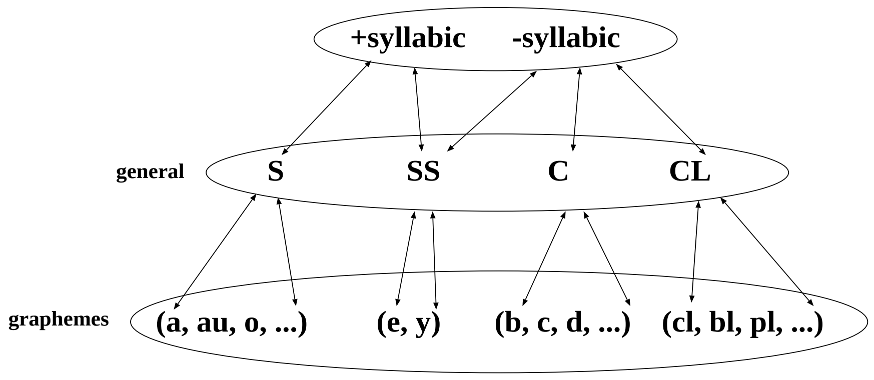


**Fig. 2.** Classification of Graphemes

Ordinary dictionaries divide syllables for hyphenation when printing rather than for pronunciation. When printing, if the last word in a line is too long, the word can be split across two lines according to dictionary hyphenation. According to the Gage Canadian Dictionary and the American Heritage Dictionary, the word *syllabic* is divided as *syl-lab-ic*. This division is not suited to pronunciation, because *syllabic* should be pronounced with the syllables *sy-lla-bic*.

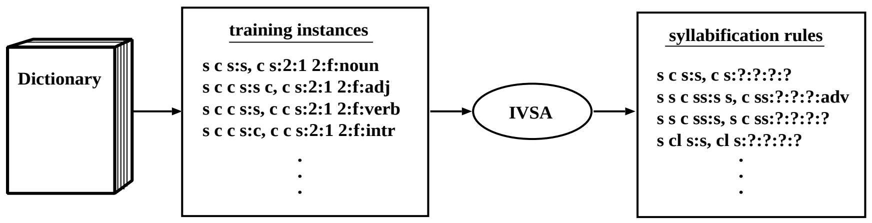
We must make clear that syllabification done by LE-SR is for pronunciation rather than printing. In this description, *syllabification* means dividing words into segments according to the pronunciation of an on-line pronouncing dictionary, while *hyphenation* means dividing words into segments according to the lexical meaning of these segments. The problem of learning hyphenation rules has been specified, using rough data, in [Dietterich, 1997]. Table 2 shows a sample of words that are both hyphenated and syllabified. Differing cuts between syllables are indicated by "·".

| Index | Word | Hyphenation | Syllabification |
|---|---|---|---|
| 1 | education | ed·u-ca-tion | e·du-ca-tion |
| 2 | dynamic | dy-nam·ic | dy-na·mic |
| 3 | folder | fold·er | fol·der |
| 4 | gravitation | grav·i-ta-tion | gra·vi-ta-tion |
| 5 | handy | hand·y | han·dy |
| 6 | infinitive | in-fin·i-tive | in-fi·ni-tive |
| 7 | jumpy | jump·y | jum·py |
| 8 | lemon | lem·on | le·mon |
| 9 | monotone | mon·o-tone | mo·no-tone |
| 10 | novel | nov·el | no·vel |
| 11 | recognize | rec·og-nize | re·cog-nize |
| 12 | shortcoming | short-com·ing | short-co·ming |
| 13 | talkative | talk·a-tive | tal·ka-tive |

**Table 2.** Hyphenated and Syllabified Words

Words are usually hyphenated at affixes such as '-ic' in "dynamic", 'grav-' in "gravitation", '-on' in "lemon", 'mon-' in "monotone", 'nov-' in "novel", 'rec-' in "recognize", and 'ing' in "shortcoming". Most cuts for hyphenation happen to be identical with those for syllabification because the graphemes next to these affixes are consonant graphemes. Otherwise, cuts for hyphenation are frequently different from those for syllabification. The hyphenations and syllabification shown in Table 2 with index numbers 1, 2, 4, 6, 8, 9, 10, 11, and 12 are such examples. Other situations causing differences between syllabification and hyphenation are compound words such as 3, 5, 7, and 13 in Table 2. These words are formed by adding suffixes '-er', '-dy', '-py', and '-tive' to root words "fold", "hand", "jump", and "talk". It makes sense to separate the root words from their suffix when printing each word on two lines, but it would violate the usual pronunciation because English speakers say "fol-der" rather than "fold-er". We believe that syllabification poses a more difficult learning problem than hyphenation.

In our approach, before syllabifying a word for pronunciation, it is first divided into graphemes, such as <ll> and <gh>. The task of grapheme segmentation is discussed elsewhere [Zhang et al., 1997]. The goal of LE-SR is to produce



Dictionary
training instances
s c s:s, c s:2:1 2:f:noun
s c c s:s c, c s:2:1 2:f:adj
s c c s:s, c c s:2:1 2:f:verb
s c c s:c, c c s:2:1 2:f:intr
IVSA
syllabification rules
s c s:s, c s:?:?:?:?
s s c ss:s s, c ss:?:?:?:adv
s s c ss:s, s c ss:?:?:?:?
s cl s:s, cl s:?:?:?:?

**Fig. 3.** Overview of LE-SR

a set of syllabification rules that can be applied to divide any English word, represented as a sequence of graphemes, into syllables. Identifying graphemes separately avoids a common difficulty with identical letter combinations that are cut in different places. For example, the word *uphill* is cut between *p* and *h* (*up·hill*), while the word *alphabet* is cut before *ph* (*al·pha·bet*). The letter combination "ph" sometimes forms one grapheme <ph> and sometimes forms two graphemes <p> and <h>. By resolving this difficulty beforehand, the problem of learning syllabification rules is simplified.

This paper does not discuss conflicting numbers of syllables for the same word due to individual usage of vocal organs (e.g., some people pronounce *sour* as one syllable and others pronounce it as two syllables). Instead, LE-SR learns a set of syllabification rules according to the NTC2 pronouncing dictionary.

## 3 The Learning Problem

Based on our previous work on learning English syllabification rules [Zhang and Hamilton, 1997], we hypothesize that this problem can be decomposed into two parts: (1) learning to identify which of the classes S, SS, C, and CL is most appropriate for each grapheme, and (2) using this information, learning to divide a sequence of graphemes into syllables. LE-SR 2.2 addresses the second problem, while assuming that the class of each grapheme has already been identified. After this information is added to input instances to create training instances, LE-SR uses the Iterated Version Space Algorithm [Hamilton and Zhang, 1996] to learn syllabification rules. As shown in Figure 3, LE-SR 2.2 first selects input instances from the on-line pronouncing dictionary and transforms them into training instances that use the C-S-CL-SS representation. These training instances are used by IVSA to generate a set of syllabification rules. The remainder of this section describes the learning problem faced by LE-SR.

**Dictionary Description:** The description of a word in the NTC2 pronouncing dictionary consists of six pieces of information: the word, the syllables of the word, the pronunciation symbols, the stress marks, a code (0 = regular, 1 = irregular, and 2 = foreign), and the parts of speech. Example:

absent [[a,b],[s,e,n,t]] [[æ,b],[s,schwa,n,t]] [[p],[n]] 0 [adj,verb,tr]

**Input:** As retrieved from the NTC2 dictionary, the input instances are 3-tuples where the first element is a word, the second element contains the graphemes that comprise the word grouped into syllables, and the third identifies the parts of speech with this syllabification. The word "dictionary" is represented as a 3-tuple:

(dictionary, [[d,i,c],[t,io],[n,a],[r,y]], [noun]).

**Creating Training Instances:** As is common with applications of machine learning, we transform the input to produce training instances that explicitly represent features we believe are relevant; this transformation introduces an inductive bias since it influences the type of rules that can be produced. In an automatic preprocessing stage, the inter-syllabic patterns in the input words are identified. An *inter-syllabic pattern* (or simply *pattern*) includes all graphemes from the first possibly syllabic grapheme (S or SS) of the current syllable up to and including the last possibly syllabic grapheme of the next syllable. For example, the word 'biology' with syllabification [[bi][o][lo][gy]] is transformed into [[C S][S][C S][C S]], which has three patterns: 'S S' (i o), 'S C S' (o l o), and 'S C S' (o g y). All patterns from the input instances are collected together.

Then the input instances are examined again and transformed into training instances. The longest possible pattern in the pattern collection is matched to the input word starting with the first possibly syllabic grapheme in the input word. A training instance is created corresponding to this match. Then the process is repeated starting with the first unmatched portion of the input word, and so on.

A training instance is created by determining the cut into syllables in the pattern, the number of syllabic graphemes in the word, the positions of all possibly syllabic graphemes of the pattern among all possibly syllabic graphemes in the word, and the rough position (first, mid, last) of the current pattern in the word. Each training instance has 6 attributes, namely Pattern, Cut, S_SS, PS, PP, and POS.

- **Pattern**: all graphemes between the first S or SS of the current syllable and the last S or SS of the next syllable, inclusive.
- **Cut**: Pattern with '-' wherever the syllables are cut
- **S_SS**: number of S and SS graphemes in the word
- **PS**: positions of the S and SS graphemes in the word
- **PP**: rough position of the pattern in the word
- **POS**: part of speech
- Example: ⟨S C S, S - C S, 2, 1 2, last, adv⟩ (aback)

**Output:** The output is a set of syllabification rules that can be used by a text-to-speech system to syllabify words. The format of the rules is the same as that of the training instances except the rules can include '?', which matches anything, instead of a specific value for any field. An example rule is:

⟨S C SS CL S, S C SS - CL S, ?, 1 2 3, last, adj⟩

This rule can be interpreted as: if the pattern 'S C SS CL S' is found at the end of an adjective, and the S and SS graphemes are located in positions 1, 2, and 3, then the syllables are cut between 'S C SS' and 'CL S'.

## 4 A Descriptive Example

Consider the following input instances:

abacus [[a],[b,a],[c,u,s]] [noun]
biological [[b,i],[o],[l,o],[g,i],[c,a,l]] [adj,noun]
calculation [[c,a,l],[c,u],[l,a],[t,io,n]] [noun]
physically [[ph,y],[s,i],[c,a],[ll,y]] [adv]

The C-S-CL-SS representations are:

abacus ⟨S C S C S C⟩ ⟨S - C S - C S C⟩
biological ⟨C S S SS S C S C S SS⟩ ⟨C S - S - SS S - C S - C S SS⟩
calculation ⟨C S SS C S SS S C S C⟩ ⟨C S SS - C S - SS S - C S C⟩
physically ⟨C S SS S C S C S⟩ ⟨C S - SS S - C S - C S⟩

LE-SR begins by collecting all patterns. From ‘abacus’, LE-SR finds the same pattern ⟨S C S⟩ twice. From ‘biological’, LE-SR saves the patterns ⟨S S⟩, ⟨S SS S⟩, ⟨SS S C S⟩, and ⟨S C S SS⟩. The word ‘calculation’ has 3 patterns, ⟨S SS C S⟩, ⟨S SS S⟩, and ⟨S C S⟩. Since the patterns ⟨S SS S⟩ and ⟨S C S⟩ have already been saved from ‘biological’ and ‘calculation’, they are not saved again. The last word ‘physically’ includes 3 patterns that are already in the pattern collection. So, they are not saved.

From the above four words, the following 12 training instances are produced:

1. ⟨S C S, S - C S, 3, 1 2, first, noun⟩ (abacus)
2. ⟨S C S, S - C S, 3, 2 3, last, noun⟩ (abacus)
3. ⟨S S, S - S, 7, 1 2, first, adj⟩ (biological)
4. ⟨S SS S, S - SS S, 7, 2 3 4, middle, adj⟩ (biological)
5. ⟨SS S C S, SS S - C S, 7, 3 4 5, middle, adj⟩ (biological)
6. ⟨S C S SS, S - C S SS, 7, 5 6 7, last, adj⟩ (biological)
7. ⟨S SS C S, S SS - C S, 6, 1 2 3, first, noun⟩ (calculation)
8. ⟨S SS S, S - SS S, 6, 3 4 5, middle, noun⟩ (calculation)
9. ⟨SS S C S, SS S - C S, 6, 4 5 6, last, noun⟩ (calculation)
10. ⟨S SS S, S - SS S, 5, 1 2 3, first, adv⟩ (physically)
11. ⟨SS S C S, SS S - C S, 5, 2 3 4, middle, adv⟩ (physically)
12. ⟨S C S, S - C S, 5, 4 5, last, adv⟩ (physically)

From these training instances, IVSA generates candidate rules. From instances 1, 2 and 12, the candidate set includes only one rule: ⟨S C S, S - C S, ?, ?, ?, ?⟩ where ‘?’ means no restriction. From number 4, 8, and 10, we have: ⟨S SS S, S - SS S, ?, ?, ?, ?⟩. Suppose we have example number 13 which is ⟨S SS S, S SS - S, 8, 3 4 5, middle, noun⟩. Then IVSA will produce another rule ⟨S SS S, S SS - S, 8, ?, ?, ?⟩ and place it before ⟨S SS S, S - SS S, ?, ?, ?, ?⟩. IVSA arranges the rules for each pattern from more specific to more general. The order of the rules is important for LE-SR because the learned rules are applied in sequence by the syllabifier.

## 5 Experimental Results

### 5.1 Ten-fold Test Results

LE-SR was tested on NTC2 [Zhang et al., 1997], a 20,000 word pronouncing dictionary for English derived from the NETalk Corpus [Sejnowski and Rosenberg, 1988]. LE-SR was applied to 90% of the input words and tested on the unseen 10% of words. Accuracy is based on the fraction of patterns that are correctly divided into syllables.

In the experiments described in Table 3, LE-SR used two sets of SS graphemes, shown as $SS_1$ = {e, y} and $SS_2$ = {e, y, l, -}. The average ten-fold testing accuracy on 10% unseen instances with $SS_1$ is 90.35% using 423 rules, while with $SS_2$ it is 89.02% using 810 rules.

| Run # | $SS_1$ = {e, y} |  |  |  |  | $SS_2$ = {e, y, l, -} |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
|  | # of Instances |  | Accuracy |  |  | # of Instances |  | Accuracy |  |  |
|  | 90% | 10% | Learning | Testing | Rules | 90% | 10% | Learning | Testing | Rules |
| 1 | 37,349 | 4,089 | 92.15% | 91.27% | 426 | 37,331 | 4,045 | 92.58% | 89.67% | 801 |
| 2 | 37,465 | 4,141 | 92.15% | 90.87% | 426 | 37,183 | 4,170 | 92.54% | 90.38% | 829 |
| 3 | 37,465 | 4,141 | 92.06% | 90.39% | 417 | 37,286 | 4,101 | 92.24% | 88.56% | 774 |
| 4 | 37,465 | 4,135 | 92.06% | 92.41% | 417 | 37,245 | 4,096 | 92.32% | 90.53% | 806 |
| 5 | 37,417 | 4,150 | 92.06% | 90.55% | 420 | 37,291 | 4,103 | 92.45% | 89.47% | 800 |
| 6 | 37,521 | 4,048 | 92.04% | 90.66% | 427 | 37,325 | 4,015 | 92.54% | 90.09% | 821 |
| 7 | 37,504 | 4,038 | 92.06% | 90.34% | 429 | 37,296 | 3,988 | 90.33% | 86.33% | 836 |
| 8 | 37,740 | 4,134 | 92.16% | 86.31% | 404 | 37,513 | 4,087 | 92.47% | 86.74% | 823 |
| 9 | 37,416 | 4,132 | 91.95% | 90.63% | 434 | 37,275 | 4,083 | 92.53% | 89.59% | 818 |
| 10 | 37,430 | 4,121 | 92.14% | 90.10% | 428 | 37,307 | 4,089 | 92.54% | 88.85% | 800 |
| Ave. | 37,466 | 4,113 | 92.08% | 90.35% | 423 | 37,305 | 4,077 | 92.25% | 89.02% | 810 |
| S.D. | 106.36 | 38.36 | 0.06 | 1.48 | 8.11 | 80.23 | 48.30 | 0.65 | 1.37 | 17.26 |

**Table 3.** Ten-Fold Test on 20,000 Words

### 5.2 Rule Usage

The frequency of rule usage (for cases where patterns were correctly classified in testing) in a single run (Run 1 with {e,y} of Table 3) is shown in Table 4. The results indicate that about 85% of the testing instances are covered by only 29 (7%) of the syllabification rules, while 157 (38%) rules are used to cover the rest of the instances. Although 231 (55%) of the rules are not used for the 10% unseen instances, they are useful for the 90% instances (see statistics in Table

4). The natural logarithms of the frequencies are also graphed in Figures 4 and 5. The rules in Figure 5 are shown in the order they will be applied. Heavy rule usage is well distributed among these rules.

| | 10% Unseen Instances | | 90% Training Instances | |
|---|---|---|---|---|
| Frequency (F) | # of Rules (%) | Usage (%) | # of Rules (%) | Usage (%) |
| F = 0 | 231 (55.40) | 0 (00.00) | 0 (00.00) | 0 (00.00) |
| F = 1 | 56 (13.43) | 56 (01.40) | 116 (27.82) | 115 (00.30) |
| F = 2 | 32 (07.67) | 64 (01.50) | 49 (11.75) | 98 (00.26) |
| F = 3 | 15 (03.60) | 45 (01.45) | 42 (10.07) | 126 (00.33) |
| F = 4 | 14 (03.36) | 56 (00.86) | 22 (05.28) | 88 (00.23) |
| F = 5 | 9 (02.16) | 45 (01.07) | 16 (03.84) | 80 (00.21) |
| $5 < F \leq 15$ | 31 (07.43) | 281 (05.40) | 61 (14.63) | 582 (01.51) |
| $15 < F \leq 35$ | 11 (02.64) | 277 (07.15) | 35 (08.39) | 912 (02.37) |
| $35 < F \leq 100$ | 12 (02.88) | 515 (12.68) | 39 (09.35) | 2,451 (06.38) |
| $100 < F \leq 1,000$ | 5 (01.20) | 1,169 (32.91) | 30 (07.19) | 9,525 (24.79) |
| $1,000 < F \leq 1,500$ | 1 (00.24) | 1,128 (35.57) | 2 (00.48) | 2,712 (07.07) |
| $1,500 < F \leq 5,000$ | 0 (00.00) | 0 | 4 (00.96) | 10,775 (28.04) |
| $5,000 < F \leq 11,000$ | 0 (00.00) | 0 | 1 (00.24) | 10,959 (28.52) |
| Total | 417 (100.00) | 3,636 (100.00) | 417 (100.00) | 38,430 (100.00) |

**Table 4.** Rule Usage Summary

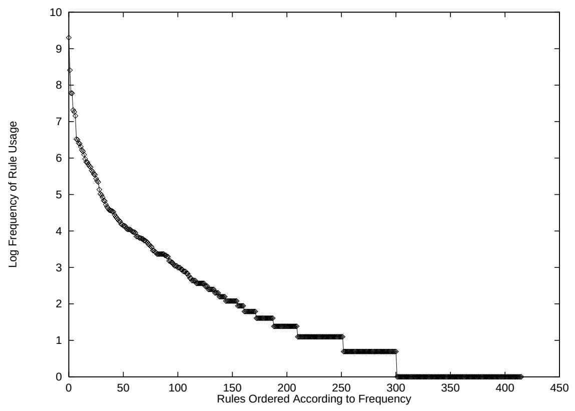


**Fig. 4.** Frequency of Rule Usage, in Descending Order

## 5.3 Experiments with Grapheme Classifications

In this section, we examine the appropriateness of the C-S-CL-SS representation introduced in Section 3. Our experiments relate to two questions: (1) should a CL

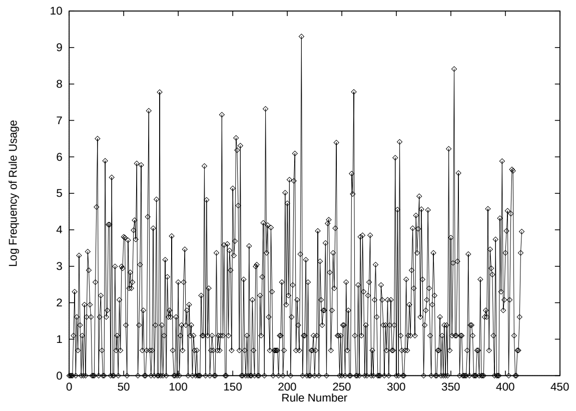


**Fig. 5.** Frequency of Rule Usage, in Rule Order

set be used, i.e., should separate CL and C sets be used or should all consonant graphemes always be treated as members of the C set? (2) which graphemes should be classified as +/− syllabic, i.e., included in the SS set?

Table 5 presents the results of preliminary experiments using different sets of graphemes in the SS set with or without using the CL set. We tested a sample of 6,000 words using six possible sets of SS graphemes. In each of the six sets of tests, the test using the CL set gave higher testing accuracy than the one with no CL set. Nonetheless, further investigation of the C-S-SS representation is warranted given the relatively high predictive accuracy in Run 4 and the small number of rules in Run 8.

| Test Number | SS Graphemes | Usage of CL Set | Accuracy Learning (15,000) | Accuracy Testing (4,200) | Number of Rules |
|---|---|---|---|---|---|
| 1 | e, i, l, u, y, -, ue | yes | 91.77% | 84.19% | 534 |
| 2 | e, i, l, u, y, -, ue | no | 89.98% | 82.37% | 545 |
| 3 | e, l, y, - | yes | 93.44% | 89.29% | 467 |
| 4 | e, l, y, - | no | 91.71% | 87.55% | 505 |
| 5 | l, y, - | yes | 90.78% | 86.71% | 308 |
| 6 | l, y, - | no | 89.01% | 84.26% | 326 |
| 7 | e, y | yes | 91.61% | 87.91% | 443 |
| 8 | e, y | no | 89.91% | 85.94% | 261 |
| 9 | l, - | yes | 90.07% | 80.73% | 280 |
| 10 | l, - | no | 88.40% | 79.15% | 286 |
| 11 | l, ue | yes | 90.15% | 86.12% | 283 |
| 12 | l, ue | no | 88.32% | 83.93% | 292 |
| 13 | none | no | 85.61% | 81.31% | 211 |

**Table 5.** Tests on LE-SR with Different Amounts of Basic Knowledge

The experiments also showed that choosing different SS sets affects both the accuracy and number of rules. However, the results do not simply improve as more graphemes are added to SS. Run 3 gives the highest learning and testing accuracy, but it does not use the most SS graphemes. Run 13 produces the fewest rules, but it also gives the lowest accuracy for both learning and testing. Our experiments suggest that whenever <e> and <y> are classified as SS, LE-SR obtains better results (Runs 1, 3, and 7).

Although the preliminary experiments on SS graphemes shown in Table 5 predicted that using set {e, y, l, -} produces highest accuracy, the ten-fold tests presented in Table 3 show that using {e, y} gives better results. Examination of the rules shows that keeping <l> as a SS grapheme all the time is not suitable. Therefore, further research on learning to classify particular occurrences of SS graphemes (such as <l>) as either syllabic or non-syllabic may yield improvements.

## 6 Conclusions

This paper has described LE-SR, the first machine learning application for learning English syllabification rules. In our ten-fold testing, LE-SR learned 423 syllabification rules from 90% of instances (average of 37,466 patterns) that gave predictive accuracy of 90.35% on the unseen 10% instances (4,120). Further experimentation with various classifications of graphemes within the C-S-CL-SS framework is warranted. Future research could also try to automate the process of identifying syllabic graphemes.

## References

[Allen et al., 1987] Allen, J., Hunnicutt, S., and Klatt, D., editors (1987). *From Text to Speech: The MITalk System.* Cambridge University Press, London.

[Dietterich, 1997] Dietterich, T. (1997). CS534 programming assignment 5. http://www.cs.orst.edu:80/ tgd/classes/534/programs/prog5/prog5.html.

[Elovitz et al., 1976] Elovitz, H., Johnson, R., Mchugh, A., and Shore, J. (1976). Automatic translation of English text to phonetics by means of letter-to-sound rules. Technical Report NRL 7948, Naval Research Laboratory, Washington, D.C.

[Hamilton and Zhang, 1996] Hamilton, H. J. and Zhang, J. (1996). The iterated version space algorithm. In *Proc. of Ninth Florida Artificial Intelligence Research Symposium (FLAIRS-96)*, pages 209–213, Daytona Beach, Florida.

[Hochberg et al., 1991] Hochberg, J., Mniszewski, S., Calleja, T., and Papcun, G. (1991). A default hierarchy for pronouncing English. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 13(9):957–964.

[Klatt, 1982] Klatt, D. (1982). The Klattalk text-to-speech system. In *Proc. Int. Conf. Acoustics Speech Signal Processing*, pages 1589–1592.

[Klatt, 1987] Klatt, D. (1987). How KLATTalk became DECtalk: An academic's experience in the business world. In *Official Proceedings Speech Tech'87: Voice Input/Output Applications Show and Conference*, pages 293–294.

[Kreidler, 1989] Kreidler, C. W. (1989). *Pronunciation of English.* Basil Blackwell, Oxford, UK.

[Ladeforged, 1982] Ladeforged, P. (1982). *A Course in Phonetics.* Harcourt Brace Jovanovich, New York.

[Ling and Wang, 1995] Ling, C. and Wang, H. (1995). A decision-tree model for reading aloud. http://www.csd.uwo.ca/faculty/ling/sub-pub.html.

[MacKay, 1987] MacKay, I. R., editor (1987). *Phonetics: the Science of Speech Production*. Pro-Ed, Austin, Texas.

[Mudambi and Schimpf, 1994] Mudambi, S. and Schimpf, J. (1994). Parallel CLP on heterogeneous networks. Technical Report ECRC-94-17, European Computer-Industry Research Centre GmbH, Munich, Germany.

[O'Grady and Dobrovolsky, 1992] O'Grady, W. and Dobrovolsky, M. (1992). *Contemporary Linguistic Analysis*. Copp Clark Pitman, Toronto.

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

[Sejnowski and Rosenberg, 1988] Sejnowski, T. and Rosenberg, C. (1988). NETtalk corpus, (am6.tar.z). ftp.cognet.ucla.edu in pub/alexis.

[Zhang and Hamilton, 1996] Zhang, J. and Hamilton, H. (1996). The LEP learning system. In *International Conference on Natural Language Processing and Industrial Applications*, pages 293–297, Moncton, New Brunswick, Canada.

[Zhang and Hamilton, 1997] Zhang, J. and Hamilton, H. (1997). Learning English syllabification for words. In *Proc. of Tenth International Symposium on Methodologies for Intelligent Systems*, pages 177–186, Charlotte, North Carolina.

[Zhang et al., 1997] Zhang, J., Hamilton, H., and Galloway, B. (September, 1997). English graphemes and their pronunciations. In *Proceedings of Pacific Association for Computational Linguistics*, pages 351–362, Ohme, Japan.