English Graphemes and their Corresponding Sound Units

Jian Zhang, Howard J. Hamilton, and Brent Galloway

Dept. of Computer Science, University of Regina
Regina, Saskatchewan, Canada, S4S 0A2
jian,hamilton@cs.uregina.ca
phone: (306) 585-4977 fax: (306) 585-4745

Abstract

The English language does not have a one-to-one correspondence between its writing system and its pronunciation system. To analyze and therefore to understand the current relationship between the writing and pronunciation systems, we propose three new definitions and present our analysis of a particular pronouncing dictionary. We also present our test results on the pronunciation rules learned from this dictionary with a machine-learning method. The analysis shows that the current relationship between the writing and the pronunciation systems can be described as a serial correspondence between sequences of graphemes and sounds if some absent graphemes and silent morphophonemes are used. We found that 26 out of 131 graphemes and 19 out of 50 sound units are used very often while the others are not. When 910 pronunciation rules learned by our computer program from one-syllable words are tested on 10% unseen examples, the predictive accuracy is 95.65% for pronouncing words and 98.79% for graphemes. Of these rules, the most used 20% of the rules cover 94% of the one-syllable words.

1 Introduction

After fifteen hundred years of development, the writing system of the English language has been left far behind its pronunciation by the evolution of the spoken language. Modern English does not have a one-to-one correspondence between letters and sounds [11]. The relationship between the English writing system and its pronunciation can be described as a serial correspondence between sequences of graphemes and sound units.

In this paper, we first define three concepts: *serial correspondence*, *absent grapheme*, and *silent morphophoneme*. Based on these definitions, a computer-readable pronouncing dictionary (*NTC2*) has been produced. Using a machine-learning approach, a set of English pronunciation rules are

learned from *NTC2*. *NTC2*'s prototype is the NetTalk Corpus (*NTC*), the computer readable pronouncing dictionary created for the NetTalk text-to-speech system [17], which was originally developed from the Webster's Pocket Dictionary and Webster's New Collegiate Dictionary. Other online pronouncing dictionaries are available, but they lack the information needed for separating graphemes.

NTC has been used in testing text-to-speech systems [8], but it apparently has not been analyzed in detail. We recognize that statistics obtained from a dictionary differ from those based on texts. Nonetheless, by analyzing a dictionary, we can identify a complete set of graphemes with their corresponding sounds for the English language. The analysis of the frequencies of these graphemes and sounds helps us to describe the complex system of English orthography. All usage results reported are with respect to the dictionary instead of with respect to text corpora.

In Section 3, an exhaustive set of graphemes and their corresponding sound units abstracted from *NTC2* are presented and analyzed. In Section 4, we present the LE-PG learning method used in this paper. In Section 5, we give a detailed example to show how the LE-PG algorithm works and why our algorithm can learn disjunctive hypotheses. In Section 6, we analyze a set of pronunciation rules learned from the one-syllable words in *NTC2* using the LE-PG learning method. Last, in Section 7, we present our conclusions and future research directions.

2 Definitions

Any English word is represented by a sequence of written symbols called *graphemes*, and the corresponding pronunciation is represented by a sequence of sound symbols called *sound units*. A *grapheme* is a letter or combination of letters that represents one phoneme, diphthong (complex vowel sound, e.g. /ei/), or cluster (complex consonant sound, e.g. /ks/). A *sound unit* (*SU*) is a general name for a phoneme, diphthong, cluster, or silent morphophoneme. A *phoneme* can be defined as a class of sounds which are phonetically similar and are predictable in complementary distribution or free variation in the phonetic environment [2]; a phoneme is the smallest units of speech which distinguishes sounds and meanings in a word [14] or as a minimum unit of distinctive sound-feature [1]. For example, the phonemes /æ/ and /Λ/ distinguish the words *c<u>a</u>t* and *c<u>u</u>t*. Phonemes have predictable variants in pronunciation, depending on the nature of the sounds that precede or follow them in a word, or the stress that occurs on them. These predictable variants are called *allophones* [15]. For example, the phoneme /æ/ has a nasal pronunciation (that is, a nasal allophone) when it is followed or preceded by nasal consonants, as in *m<u>a</u>n*, where the /æ/ is nasal. Each time a phoneme is realized (through its allophones) in an actual spoken utterance, we call it a *phone*. We represent a grapheme in angle brackets, a phoneme between two slashes, and a phone or allophone between square brackets.

There is no one-to-one correspondence between the set of English letters and the set of English sound symbols. The basic units in the writing system which do correspond to individual sounds are graphemes rather than letters. To present our analysis of the relationship between English graphemes and their pronunciation, we first define the concept of *serial correspondence*.

Definition 1 *A* serial correspondence *between the sequence of graphemes in a word and the sequence of sound units in its pronunciation exists if the following conditions are satisfied:*

1. *The number of graphemes in a word is equal to the number of the sound units in its pronunciation.*
2. *The first grapheme in a word corresponds to the first sound unit in its pronunciation, the second grapheme in a word corresponds to the second sound unit in its pronunciation, and so on.*

The serial correspondence between the word *thesis*, with graphemes <th,e,s,i,s> and its pronunciation /θ,i:,s,i,s/, is that the first grapheme <th> corresponds to the first sound unit /θ/, the second grapheme <e> corresponds to the second sound unit /i:/, and so on.

Consider the word *Mccoy* with graphemes <M,cc,oy> and pronunciation /m,∂,k,oi/: no grapheme corresponds to the /∂/ sound at all. On the other hand, some graphemes, such as <gh> in the words *night*, *bright*, and *sigh*, are not pronounced. To allow a serial correspondence, we define the concepts *absent grapheme* and the *silent morphophoneme*.

Definition 2 *If there is an epenthetic sound that cannot be conveniently represented by a grapheme, we say that there exists an* absent grapheme*, which we represent by* <–>.

Thus, for *McCoy*, we have <M,–,cc,oy> and for *Mccarthyism*, we have <M,–,cc,a,r,th,y,i,s,m>. Here the absent grapheme, <–>, represents (as it often does) an epenthetic schwa phoneme.

A silent letter may be treated individually as a silent grapheme or incorporated into a neighboring grapheme. Suppose the silent <gh> is treated as part of a neighboring grapheme. Consider group 1 {weigh, plough, high, aught} and group 2 {tough, laugh, enough, cough}. For the four words in group 1, we could propose the graphemes <eigh>, <ough>, <igh>, and <augh>. This is acceptable but requires four new graphemes. This approach is awkward for the words in group 2, where <gh> is best treated as a grapheme that is pronounced with an /f/ sound. Therefore, we adopt a simpler approach where <gh> is treated as a grapheme that is sometimes pronounced and sometimes silent.

Definition 3 *A "silent" morphophoneme occurs when a grapheme is not pronounced. The symbol* [–] *represents a silent morphophoneme.*

Other examples of silent graphemes are <e> in *ache*, <w> in *two*, <ch> in *yachtsman*, and <l> in *should*.

3 The *NTC2* Pronouncing Dictionary

The *NTC2* pronouncing dictionary is based on the *NTC* pronouncing dictionary adjusted to ensure serial correspondence between graphemes and sound units. In this section, we describe and analyze this dictionary to illustrate the scale of the problem of learning pronunciation rules. The analysis presented here depends only on the dictionary; it is independent of the machine learning approach described in subsequent sections.

The *NTC2* pronouncing dictionary contains 20,000 English words, (including 3723 one-syllable words), 131 graphemes, and 50 *SU*s. It is essentially a reformatted version of *NTC* with some errors corrected and some inconsistencies resolved. Table 1 shows the number of sound units that each

	Number of SUs that One Grapheme represents											Total
	1	2	3	4	5	6	7	8	10	11	14	
Number of Graphemes	56	22	15	15	9	6	3	3	1	1	1	132
Total Sound Units	56	44	45	60	45	36	21	24	10	11	14	366

Table 1: Number of *SU*s that One Grapheme Represents

grapheme can represent, how many graphemes of each group have the same number of sound units, and the total number of sound units shown by each group. The first column shows that 56 graphemes represent unique *SU*s; the second column shows that 22 graphemes represent 2 different *SU*s each; and the third column shows that 15 graphemes represent 3 different *SU*s each, etc. Interestingly, the number of graphemes decreases as the number of *SU*s that one grapheme can represent increases. That is, 60% of the graphemes represent one or two *SU*s. The vowels <o> (with 14 SUs), <a> (with 10 SUs), <u> (with 10 SUs), <e> (with 9 SUs), and <i> (with 8 SUs) are among the most frequently occurring graphemes in *NTC2*. Only 3 graphemes represent 10 or more *SU*s each. The graphemes <a>, <e>, and <o> require the most time to learn because many SUs are involved, many instances are available, and many inconsistent pronunciation exist among the instances. Our results complement those of [5], who studied the number of graphemes per phoneme for a text corpus of 17,310 words.

Figure 1 shows the frequencies of graphemes in *NTC2* in descending order of frequency. Only about 20 of the graphemes have frequencies above 1,000, and more than 100 graphemes have frequencies below 500. In [10], 45 letter sequences that might be graphemes are identified and ranked according to three factors: frequency of occurrence, length in letters, and frequency of cases where the letter sequence corresponds to more than one sound. The result was a ranked list of 45 letter sequences, but apparently the actual usage of these letter sequences was not studied.

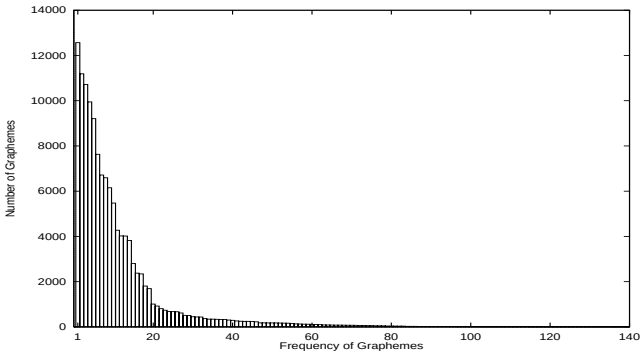

Figure 1: Frequency of Graphemes in *NTC2*

Figure 2 shows the cumulative frequency of the graphemes in *NTC2*. The hyperbolic graph shows that the frequencies of the graphemes follow the Bradford-Zipf distribution. The Bradford-Zipf distribution is a combination of Bradford's law of scattering and Zipf's law of relationship between the rank of an item and the frequency of the item [7]. Bradford's law concentrates on a small number of items which have high frequencies, but Zipf's law concentrates on a large number of items which have low frequencies. Instead of focusing on either the lower or higher range, the Bradford-Zipf distribution describes both.

A sample of items is said to fit the

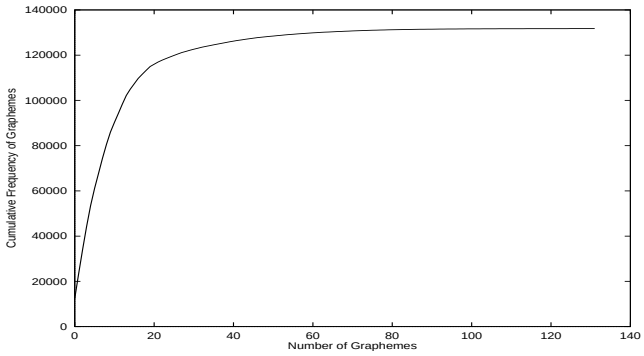

Figure 2: Cumulative Frequency of Graphemes in *NTC2*

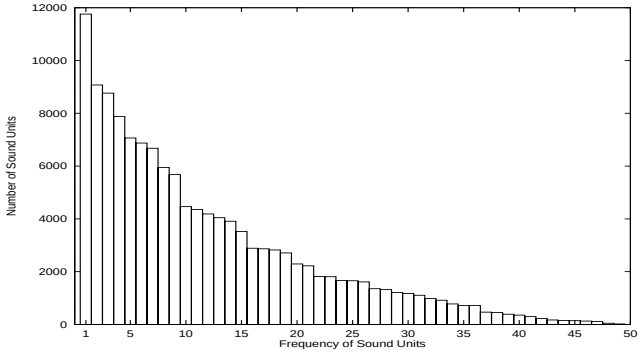

Figure 3: Frequencies of the Sound Units in *NTC2*

Bradford-Zipf distribution if the importance of items in the sample diminishes rapidly as more items are considered. Usually, there are only a few items that are very important and a large number of items that are not so important. Once the few important items are identified, the rest can be ignored in many cases. A typical Bradford-Zipf distribution approximates the "80-20 rule" given in [9]. Applying this rule to our problem, we interpret it as: 20% of the graphemes account for at least 80% of grapheme usage. As shown in Figure 2, 20% of the graphemes (26) occur 91% of the time (119,871 out of 131,746 occurrences of graphemes in the *NTC2* pronouncing dictionary). Therefore, grapheme usage follows the "80-20 rule" and matches the Bradford-Zipf distribution.

In *NTC2*, the phoneme /∂/ has the greatest frequency, with 11,963 occurrences, and the cluster /k_sh/ has the lowest frequency, with 18 occurrences.

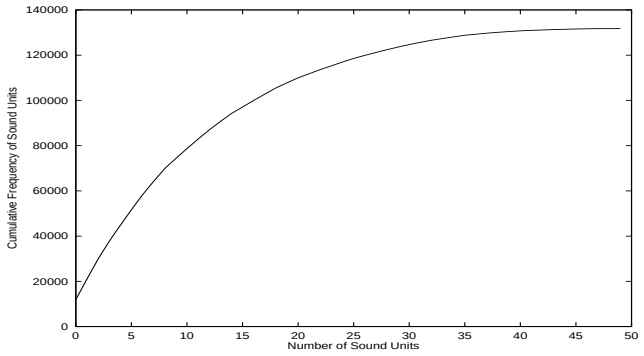

Figure 4: Cumulative Frequency of Sound Units in *NTC2*

Figure 3 present a graph showing the frequencies of *SU*s in *NTC2*. Interestingly, the usage of *SU*s is less concentrated than that of graphemes. About 20% of them are above 5,000 in frequency, 40% are above 1,000, and only 6% of them are lower than 50.

Figure 4 shows the cumulative frequency for the 50 *SU*s in *NTC2* with a graph. The most commonly used 20% (10) of the *SU*s have a cumulative frequency of 74,338, which shows that these 20% of graphemes have 56% of the frequency in *NTC2*. Apparently, the usage of *SU*s does not follow the 80-20 rule, but it still follows the Bradford-Zipf distribution because the 39% most commonly used *SU*s (19 *SU*s) are accessed 80% of the time in *NTC2*.

4 The *LE-PG* Component

Using a machine-learning approach [12], based on the Version Space Algorithm [13], a set of computer programs was designed to obtain a set of complete pronunciation rules for the orthography of all (3,723) one-syllable words in *NTC2*. This set of computer programs is called *LE-PG* [18], which stands for "Learning English Pronunciation for Graphemes".

Input: I, a set of input instances
Output: H, an ordered list of regional hypotheses
$I' := I$, $H := \langle\ \rangle$, $Acc_0 := 0$
repeat for $i = 1, 2, ...$
    $CH :=$ Generator(I')
    $(H, Acc_i) :=$ Assembler(I, H, Acc_{i-1}, CH)
    $I' :=$ Incorrect(I, H)
until $I' = \phi$ or $(i > max$ **and** $Acc_i = Acc_{i-max})$

Figure 5: The IVSA Algorithm

The goal of *LE-PG* is to obtain a set of reliable pronunciation rules for English graphemes. We assume that pronunciation is specified in terms of the International Phonetic Alphabet (IPA) [6]. The structure of *LE-PG* is based on the Iterated Version Space Algorithm (IVSA) [4] [19] [20] [21].

In Figure 5, we present the IVSA algorithm. The main idea is to find regional hypotheses (RHs) of a learning concept, where a regional hypothsis is an ordered disjunctive description of a concept. First, IVSA uses the Generator algorithm to produce a set of candidate hypothese from the set I' of input instances that are not yet covered. Then, the Assembler selects the most promising candidate hypotheses and orders them to form a single disjunctive hypothesis.

The Generator algorithm (Figure 6) handles multiple values of the decision attribute. VSA requires that instances be classified as positive and negative. For each decision value, we create a set X of appropriate instances by marking all instances with this decision value as positive and all others as negative. (In our implementation, we actually use a variant of VSA called MVSA [3], which obviates the need for this translation.) Each iteration of the inner loop generates multiple candidate RHs for a single decision value. Each RH is consistent with a series of instances. If an instance x_m is encountered that (according to VSA) would destroy the version space for X, the Generator instead stores the existing specific hypothesis set (S) and the existing general hypothesis set (G) and then begins constructing a new RH, using the instances beginning with x_m as input. Thus, each RH is constructed from mutually consistent instances. Finding inconsistent instances of a concept provides a natural way of separating regions of a concept; these regions form the basis for a disjunctive concept hypothesis.

The Assembler algorithm (Figure 7) first ranks the candidate RHs produced by the Generator. Ranking is done according to measure $R = (|P_C| + |N_C|)\ /\ |I|$, where P_c is the set of positive instances that correctly match the hypothesis, N_c is the set of negative instances that correctly do not match the hypothesis, and I is the complete set of instances. R indicates the quality of an RH.

The Assembler then considers whether each candidate RH h should be added to the list H of accepted hypotheses, as follows. First h is added to H to form H'. If the classification accuracy is higher with H' than with H, then h is kept in the accepted list H. If other hypotheses are already present in H, then h is tested in all positions in the list, and the position resulting in the highest accuracy is selected. This process is repeated for each candidate RH. Given c candidate RHs and d accepted hypotheses, the Assembler performs $O(c(c + d))$ tests. Finally, the Assembler considers whether any accepted hypotheses could now be deleted because of other hypotheses that have been inserted in the list of accepted hypotheses.

5 A Descriptive Example

We now present an application of IVSA to learning rules for pronouncing English graphemes. The ordered list of rules for pronouncing a single grapheme is treated as

```
Input: I', a set of uncovered input instances
Output: CH, a set of candidate regional hypotheses
CH := φ
repeat for each possible decision value
    X := instances from I' classified neg or pos
    repeat until X = φ
        (S, G, x_m) := VSA(X)
        X := X − {x_1, ..., x_{m−1}}
        CH := CH ∪ S ∪ G
    end
end
```

Figure 6: The Generator Algorithm

```
Input: I, a set of input instances;
       H, the initial list of accepted RHs;
       Acc, the initial classification accuracy;
       CH, a set of candidate RHs
Output: updated H and Acc

/* Insert new rules to existing rules */
/* in an appropriate position */
for each candidate hypothesis h_i ∈ CH
    R_i := (|P_C| + |N_C|) / |I|
end
for h ∈ CH, ordered by decreasing R values
    BestAcc' := 0
    for j = 1, 2, ...|H| + 1
        H' := H with h inserted before position j
        Acc' := |Correct(I, H')| / |I|
        if Acc' > BestAcc' then
            BestH' := H', BestAcc' := Acc'
        end if
    end for
    if BestAcc' > Acc then
        H := BestH', Acc := BestAcc'
    end if
end for

/* Remove unnecessary hypotheses from H */
for h ∈ H
    H' := H − {h}
    Acc' := |Correct(I, H')| / |I|
    if Acc' ≥ Acc then
        H := H', Acc := Acc'
    end if
end for
```

Figure 7: The Assembler Algorithm

a single disjunctive concept. IVSA is applied separately for each grapheme. Here, we consider the grapheme <a>, which has very complicated pronunciation rules. The input instances are drawn from *NTC2*, and pronunciation is specified according to the IPA. Both the input instances and the regional hypotheses have 10 attributes (Table 5). For a RH, '?' may appear as a value for any attribute to indicate that no restriction is placed on its value.

Each input instance describes one occurrence, in a word, of the grapheme <a>, including its IPA sound symbol. In *NTC2*, the grapheme <a> happens to correspond to 10 different sound symbols (10 values for the decision attribute). A positive instance contains the target grapheme <a> and the target sound symbol, while a negative instance contains the target grapheme but not the target sound symbol. We let the sound symbol /æ/ be first learning target; i.e., instances with /æ/ are positive and instances with the other 9 sound symbols are negative. Two instances from the word *abacus* are:

(ash, p, open, empty, empty, b, a, a, a, 3)

(schwa, n, open, a, b, c, u, b, a, 3)

where 'ash' represents /æ/. While learning rules for /æ/, the first instance is treated as positive and the second as negative.

We denote the individual general hypotheses as $G_1, G_2, \ldots, G_j$, according to the order in which they are generated by VSA. Similarly, we denote the individual specific hypotheses as $S_1, S_2, \ldots, S_k$. For example, in Figure 8, the initial G set contains one hypothesis G_1 = (ash, ?, ?, ?, ?, ?, ?, ?, ?, ?), and the initial S set contains S_1 = (ash, n, closed, empty, c, l, c, c, l, 2). Positive instances are denoted as $P_1, P_2, \ldots, P_p$, and negative ones as $N_1, N_2, \ldots, N_n$.

The Generator does not include all positive instances and exclude all negative instances in one step. Instead, an alternating

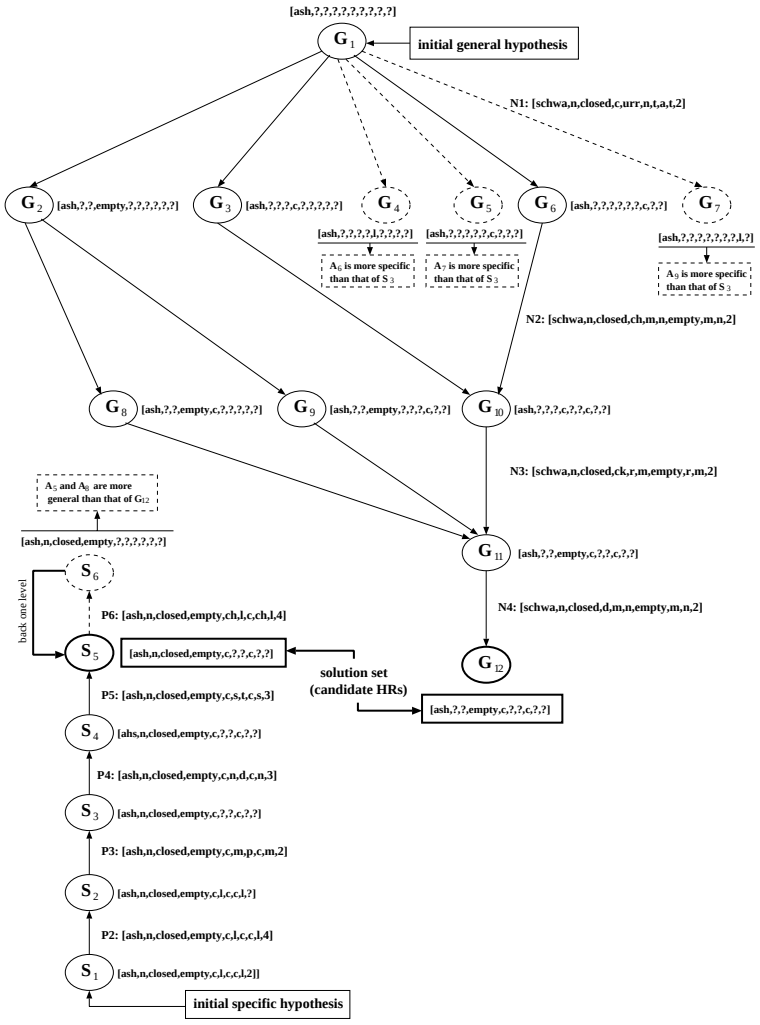

Figure 8: Hypothesis Space after One Iteration

sequence of positive and negative instances is examined by the IVSA algorithm. Figure 8 shows an example of one iteration of the inner loop of the Generator. When positive instance P_6 = (ash, n, closed, empty, ch, l, c, ch, l, 4) is examined, IVSA produces a new specific hypothesis S_6, (ash, n, closed, empty, ?, ?, ?, ?, ?, ?). Since the current general hypothesis G_{12} is (ash, ?, ?, empty, c, ?, ?, c, ?, ?), S_6 is overly generalized in A_5 and A_8. By the definition of VSA, S_6 is pruned, which produces an empty S set. An empty hypothesis set indicates an inconsistency in the instances and VSA ordinarily would find no solution. Instead of failing, IVSA takes the previous S set, which here is S_5, as a possible partial solution. In this case, hypotheses S_5 and G_{12} correctly describe all input instances from when VSA was started until P_6 was encountered. Thus, S_5 and G_{12} are candidate RHs.

6 Pronunciation Rules

For our first experiments, the input was 3,724 one-syllable words from *NTC2*. These words include 17,723 unique grapheme examples, 98 different graphemes, and 46 different sound units.

Attr.	Purpose	Example Values
A_1	sound unit	/æ/, /gz/, /ei/
A_2	stress on a syllable	p, s, n
A_3	type of a syllable	open, closed
A_4	second grapheme before target	any grapheme
A_5	first grapheme before target	any grapheme
A_6	first grapheme after target	any grapheme
A_7	second grapheme after target	any grapheme
A_8	first grapheme of the syllable	any grapheme
A_9	last grapheme of the syllable	any grapheme
A_{10}	# of syllables	1, 2, 3, 4

Table 2: Attributes in Input Instances

Rules Learned from 90% Input Instances		
Run	Correct Words	Correct Graphemes
1	96.51%	98.86%
2	95.16%	98.74%
3	97.85%	99.43%
4	95.16%	98.71%
5	96.24%	98.85%
6	93.01%	98.12%
7	96.24%	98.99%
8	94.64%	98.50%
9	95.71%	98.83%
10	95.98%	98.87%
Average	95.65%	98.79%

Table 3: Average of Ten-fold Test Result:

LE-PG was applied to 90% of the input instances (positive and negative) and tested on the unseen 10% instances (positive and negative) in ten separate runs (see Table 3). For the experimental run k, the unseen instances were $\{x_j \mid j \ mod \ 10 = k\}$, i.e., x_1, x_{11}, x_{21}, ... for run 1. *LE-PG* learned an average set of 910 rules for one-syllable word pronunciation. Correct IPA pronunciations were produced for an average of 95.65% of the unseen words and an average of 98.79% of the graphemes in the unseen words. For example, if the word *abacus* is translated as /æ, b, ei, k, ∂, s/ instead of /æ, b, ∂, k, ∂, s/, then 5 out of 6 graphemes are translated correctly, but 0 words are translated correctly. The performance of IVSA on this problem has been compared to that of C4.5 [16]. IVSA obtained higher accuracy using more rules; for detailed results see [20].

The rules learned from the one-syllable words were then tested on multi-syllable words. Table 4 shows the rule usage for the 35 most frequently used rules when they were applied to all of the words in *NTC2*. The usages given represent only the cases where correct pronunciations resulted. The rule usages are classified by the number of syllables in the words translated; the number of incorrect pronunciations are summarized in the last line of Table 4. In general, the same 35 rules are used most frequently for translating both one-syllable and multi-syllable words. For example, the first rule in Table 4 is used 913 times for correctly translating one-syllable words, and it is also used 2,628 times for two-syllable words, 2,454 times for three-syllable words, 1,879 times for four-syllable words, and 873 times for words with more than four syllables. Overall statistics show that the 35 most frequently used rules are used 75% of the time for correctly translating one-syllable words, 87% for two-syllable words, 92% for three-syllable words, 94% for four-syllable words, and 96% for words with more than four syllables. Although the pronunciation rules for one-syllable words do not cover all features of multi-syllable words, it is clear that the most useful set of rules for one-syllable words is also very useful for multi-syllable words.

LE-PG learned 910 pronunciation rules when applied to one-syllable words in *NTC2*. Table 5 shows the classified frequencies for these rules. Of the 910 rules, 28 (3%) are used more than 100 times, but 674 (74%) are used only one time. Of these 28

Grapheme	SU	Frequency of Rule Usage					
		1-syllable	2-syllable	3-syllable	4-syllable	5+ syllable	Total
t	t	913	2628	2454	1879	873	8747
n	n	583	2326	2613	2012	973	8507
r	r	937	2365	2061	1522	715	7600
i	i	370	1334	1270	1249	698	4921
s	s	750	1503	1123	687	348	4411
l	l	782	1242	941	747	478	4190
m	m	382	1242	1229	889	388	4130
c	k	238	1077	1251	994	466	4026
p	p	557	1352	1060	725	314	4008
e	–	816	1550	982	466	151	3965
d	d	507	1464	1093	645	238	3947
b	b	396	979	673	459	297	2804
0	o	177	610	529	304	203	1823
v	v	156	542	593	395	118	1804
er	schwa_r	34	723	628	299	88	1772
f	f	309	626	429	222	100	1686
e	e	241	548	399	283	97	1568
g	*g*	256	521	382	238	85	1482
a	æ	336	563	325	136	25	1385
a	ei	173	317	245	446	199	1380
e	e	28	195	453	377	148	1201
u	inv_v	259	328	211	78	41	917
a	æ	2	200	207	244	157	810
k	k	269	333	109	22	4	737
i	ai	149	271	198	66	16	700
w	w	200	328	131	13	2	674
ll	l	93	242	161	109	43	648
ss	s	46	213	105	49	29	442
sh	sh	145	223	55	11	2	436
a	a	49	73	143	73	80	418
a	a	68	166	109	49	25	417
h	h	119	162	62	40	19	402
o	open_o	33	148	125	62	21	389
ng	eng	61	224	67	17	3	372
x	ks	33	123	95	85	34	370
Covered Correctly		10,467	26,741	22,511	15,892	7,478	83,089
Covered Incorrectly		3,420	4,141	2,081	1,030	331	11,103

Table 4: Rule Usage for the Most Frequently Used Rules

Classified Frequencies (f)	Number of Rules
$f > 100$	28
$100 \geq f > 30$	34
$30 \geq f > 5$	48
$5 \geq f > 2$	55
$f = 2$	71
$f = 1$	674
Total Number of Rules	910

Table 5: Classified Frequencies of Rule Usage (One-syllable Words)

rules, 27 are also heavily used for translating multi-syllable words and are present in in Table 4. The 674 rules are used to deal with the special cases not covered by the more general rules, i.e., exceptions to the usual English pronunciation. These results suggest that the principal characteristics of English pronunciation are covered by 3% of the pronunciation rules.

Figure 9 shows the cumulative frequency of the rules' usage, for multi-syllable words, which again follows the Bradford-Zipf distribution. The most frequently used 20% (182)

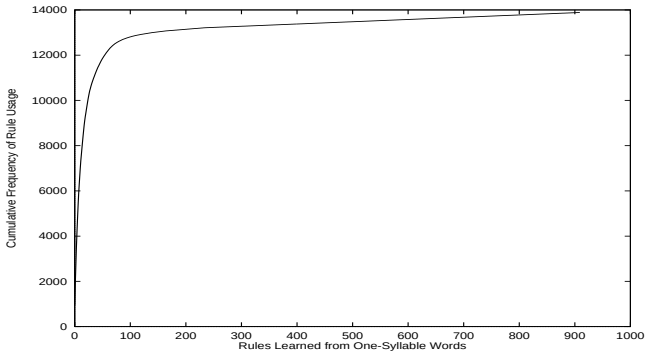

Figure 9: Cumulative Frequency of Rule Usage

of the rules have a cumulative frequency of 13,106, which is 94% of the total. Thus, rule usage follows the 80-20 rule explained in Section 3. The basic structure of English pronunciation is described by 20% of the pronunciation rules.

7 Conclusions

To construct a computer readable pronouncing dictionary, three new definitions were proposed. A serial correspondence was defined between written forms and sound units such that every grapheme in an English word corresponds to a sound unit in the pronunciation of the word. To allow serial correspondence in exceptional cases, the absent grapheme <–> and silent morphophoneme [–] were defined to remediate the missing graphemes and silent sounds.

Based on this serial correspondence, a computer readable pronouncing dictionary *NTC2* was produced. There are 131 graphemes and 50 *SU*s in *NTC2*. A set of highly reliable pronunciation rules were learned from 90% of the input examples of one-syllable words. When tested on the remaining 10% of unseen examples, these rules have an accuracy of 95.65% at word level and 98.79% at the grapheme level. The cumulative frequencies of the usage of graphemes, *SU*s, and pronunciation rules all follow the Bradford-Zipf distribution. That is, 20% of the total graphemes, 39% of the total *SU*s, and 20% of the total pronunciation rules are the most important and the rest of the graphemes, *SU*s, and pronunciation rules are much less important.

By analyzing and reconstructing the pronouncing dictionary, we have gained additional insight into the relationship between English graphemes and their pronunciations. Although the English writing system does not have a one-to-one correspondence between letters and sounds, our way of defining graphemes and sound units has enabled us to identify a serial correspondence between English graphemes and their pronunciations. This correspondence makes machine learning of English pronunciation rules possible.

The *LE-PG* learning programs produced 910 pronunciation rules including exceptions. Of the 910 rules, only 28 are used more than 100 times during the translation of one-syllable words in *NTC2*, while 674 are used only one time. These 28 rules are also intensively used for translating multi-syllable words. These statistics suggest that the main characteristics of English pronunciation are concentrated in a small number of pronunciation rules, while the majority of pronunciation rules are seldom used.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada, the Institute for Robotics and Intelligent Systems, and the University of Regina.

References

[1] L. Bloomfield. *Language*. Holt, Rinehart, and Winston, New York, 1933.

[2] D. Crystal. *A Dictionary of Linguistics and Phonetics*. Basil Blackwell Inc., Cambridge, 1985.

[3] H. J. Hamilton and J. Zhang. Learning pronunciation rules for English graphemes using the Version Space Algorithm. In *Proc. of Seventh Florida Artificial Intelligence Research Symposium (FLAIRS-94)*, pages 76–80, Pensacola Beach, FL, May 1994.

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

[5] P.R. Hanna and J.S. Hanna. *Phoneme-Grapheme Correspondences as Cues to Spelling Improvement*. U.S. Department of Health, Education, and Welfare, 1966.

[6] D. Jones and A.C. Gimson. *Everyman's English Pronouncing Dictionary*. J.M. Dent, London, 1981.

[7] A. Kent, H. Lancour, and J. Daily. *Encyclopedia of Library and Information Science*. Marcel Dekker, New York, 1979.

[8] D.H. Klatt. Review of text-to-speech conversion for English. *Journal of the Acoustical Society of America*, 82:737–793, 1987.

[9] D. E. Knuth. *The Art of Computer Programming: Searching and Sorting*. Addison-Wesley, Reading, Mass, 1973.

[10] C. Ling and H. Wang. A decision-tree model for reading aloud. Internet server: ftp.csd.uwo.ca/pub/ling/papers/-mlj-bs.ps.Z, 1995.

[11] I.R.A. Mackay. *Phonetics: The Science of Speech Production*. Pro.Ed, Austin, Texas, 1987.

[12] R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors. *Machine Learning: An Artificial Intelligence Approach*. Tioga Publishing Company, Palo Alto, CA, 1983.

[13] T.M. Mitchell. *Version Spaces: An Approach to Concept Learning*. PhD thesis, Stanford University, CA, 1978.

[14] I. Morris, editor. *The American Heritage Dictionary*. Houghton Mifflin, Boston, MA, 1991.

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

[16] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.

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

[18] J. Zhang. Automatic learning of English pronunciation rules. Master's thesis, University of Regina, Regina, Canada, 1995.

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

[20] J. Zhang and H.J. Hamilton. Learning English pronunciation rules: A comparison of IVSA and C4.5. In *Proc. of Tenth Florida Artificial Intelligence Research Symposium (FLAIRS-97)*, pages 209–213, Daytona Beach, Florida, May 1997.

[21] J. Zhang and H.J. Hamilton. Learning English syllabification for words. In *Proc. of Tenth International Symposium on Methodologies for Interlligent Systems*, Charlotte, North Carolina, October 1997. To appear.